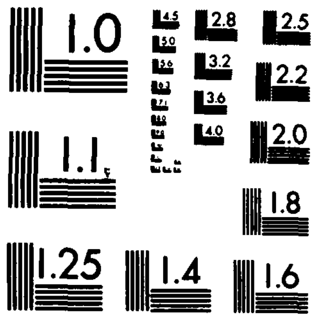
AD-A128 719 STUDY OF A HETEROGENEOUS DISTRIBUTED MICROCOMPUTER NETWORK USING MEASURED..(U) GEORGIA INST OF TECH ATLANTA SCHOOL OF ELECTRICAL ENGINEERING.. J L HAMMOND ET AL. MAR 83 E21-616 1/1

UNCLASSIFIED F/G 9/2 NL

END
DATE
FILMED
DTIC


1.0
4.5
5.0
5.6
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

15

# AIRMICS

US ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION AND COMPUTER SCIENCE

COMPUTER SYSTEMS COMMAND
INFORMATION FOR DECISION
UNITED STATES ARMY

# STUDY OF A HETEROGENEOUS DISTRIBUTED MICROCOMPUTER NETWORK USING MEASURED DATA AND ANALYTIC/SIMULATION MODELS

MARCH 1983

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. DAOG7247 A128719 | 3. RECIPIENT'S CATALOG NUMBER ----- |
| 4. TITLE (and Subtitle) STUDY OF A HETEROGENEOUS DISTRIBUTED MICROCOMPUTER NETWORK USING MEASURED DATA AND ANALYTIC/SIMULATION MODELS | | 5. TYPE OF REPORT & PERIOD COVERED FINAL REPORT 10 JUN 80 - 8 SEP 82 |
| | | 6. PERFORMING ORG. REPORT NUMBER E21-616 |
| 7. AUTHOR(s) Dr. Joe L. Hammond, Dr. Jay H. Schlag | | 8. CONTRACT OR GRANT NUMBER(s) DAAG29 80K0009 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS School of Electrical Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62725A SX762725DY10 05-001 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS US Army Research Office, P. O. Box 12211, Research Triangle Park, N. C. 27709 | | 12. REPORT DATE March 1983 |
| | | 13. NUMBER OF PAGES 68 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) US Army Institute for Research in Management Information and Computer Science. AIRMICS USA CSC-AT 115 O'Keefe Building 61T Atlanta, Georgia 30332 | | 15. SECURITY CLASS. (of this report) Unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE None |

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

SAME

18. SUPPLEMENTARY NOTES

Project managed by Dale N. Murray, Project Officer, AIRMICS, (404) 894-3136. This project has evolved into a facility called SPANET, for System for Performance Analysis of NETworks.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microcomputer, Networks, Emulator, Simulation, LAN, LCN, Performance Evaluation, CSMA, ETHERNET, Local Area Networks, Local Computer Networks, Analytic Model, Performance Monitoring, Performance Characteris tics, Test Bed Facility, SPANET, MLCN

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The objective of this project, initiated by the U. S. Army Institute for Research in Management Information and Computer Science was to develop a cost-effective alternative to discrete-event simulation methods for studying carrier sense multiple access networks. This report contains sections on the design, conceptual approach, and implementation of the CSMA emulation facility and preliminary experiments performed to validate it.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This work was done under Contract Number DAAG29-80-K-0009 from the U. S. Army Research Office for the United States Army Institute for Research in Management Information and Computer Sciences (AIRMICS), a part of the United States Army Computer Systems Command. This report is not to be construed as an official Department of the Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited. Not protected by copyright laws.

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED.

Clarence Giese
Clarence Giese, Director
U. S. Army Institute for Research
in Management Information and
Computer Sciences

[illegible]
John R. Mitchell, Chief
Advanced Information
Systems Division
AIRMICS

Dale N. Murray
Dale N. Murray, Project Officer
Advanced Information Systems Division
AIRMICS

DTIC COPY INSPECTED 2

| Accession For | |
|---|---|
| NTIS GRA&I | ✓ |
| DTIC TAB | ☐ |
| Unannounced | ☐ |
| Justification | |
| By | |
| Distribution/ | |
| Availability Codes | |
| Dist | Avail and/or Special |
| A | |

# STUDY OF A HETEROGENEOUS DISTRIBUTED MICROCOMPUTER NETWORK USING MEASURED DATA AND ANALYTIC/SIMULATION MODELS

Principal Investigators:

J. L. Hammond
J. H. Schlag

Contributors:

Dan K. Lam
Peter O'Reilly
Dale Murray

Submitted to

United States Army Institute for Research
in Management Information and Computer Science

March 1983

School of Electrical Engineering
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia 30332

## ABSTRACT

The objective of this project, initiated by the U.S. Army Institute for Research in Management Information and Computer Science, was to develop a cost-effective alternative to discrete-event simulation methods for studying Carrier Sense Multiple Access networks. This report contains sections on the design, conceptual approach, and implementation of the CSMA emulation facility and preliminary experiments performed to validate it.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

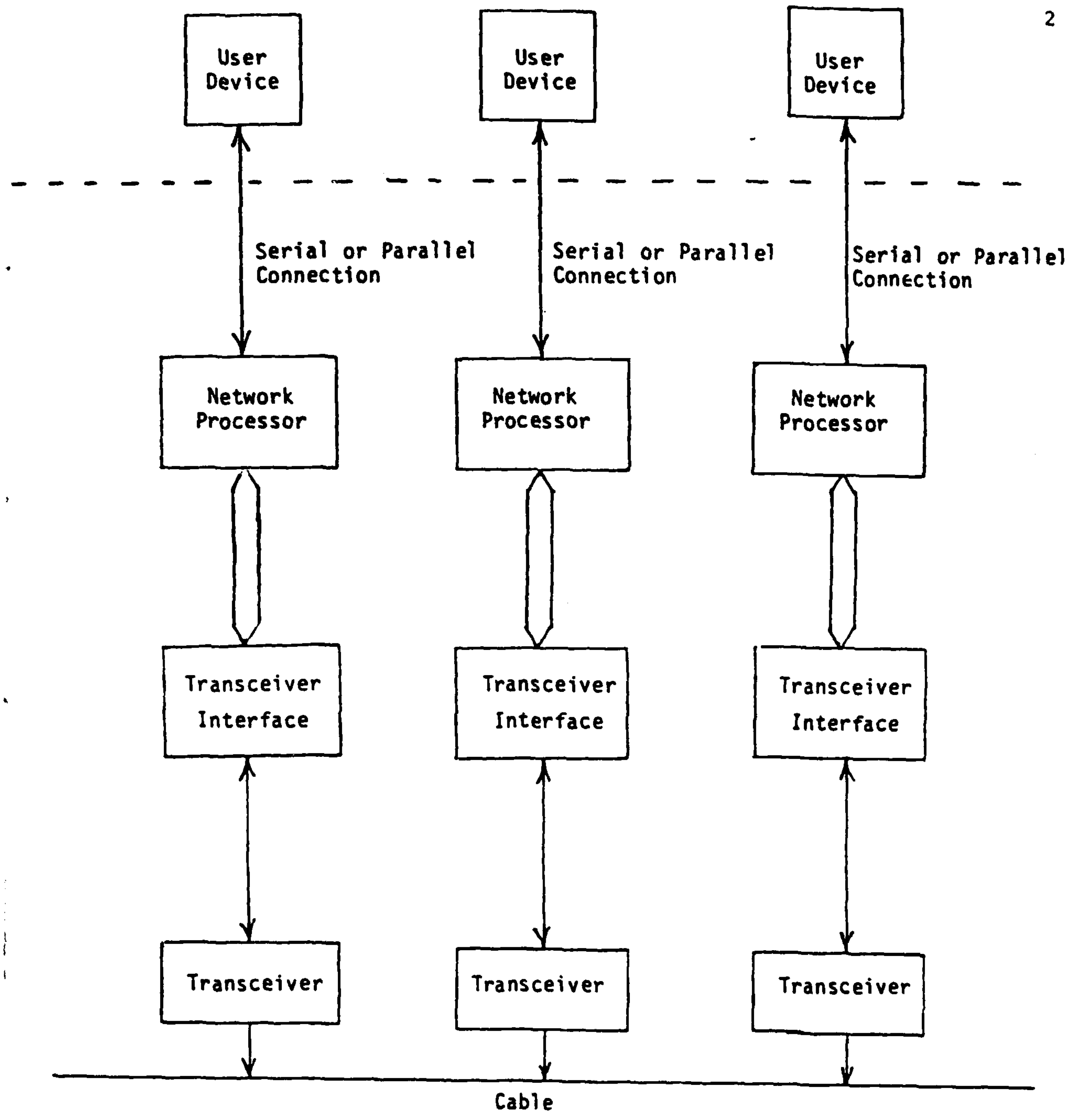
# I. INTRODUCTION

In the last five years, there has been an increasing interest in local area networks and in the Carrier Sense Multiple Access Collision Detection (CSMA/CD) type specifically. Commercial networks of this type are becoming available and include such networks as Ethernet, Net One and Want Net, [MOKH 82]. Ethernet is the most widely documented, and a detailed description is included as an appendix in the book by Franta and Chlamtac, [FRAN 81]. Most of the networks listed claim the ability to support many hundreds of stations. For example, Ethernet claims to support on the order of one thousand.

A typical CSMA/CD type local computer network has the general structure shown in Figure 1.1. Note that each station has three major components: the Network Processor, the Transceiver Interface and the Transceiver. In the notation used for the Ethernet architecture, the Network Processor supports the Data Link Layer, while the Transceiver Interface and the Transceiver support the Physical Layer, as identified in the ISO model (see, for example, [TANE 81]).

Performance studies of CSMA-type networks can obviously be carried out using the brute force of building a testbed with a number of different networks deployed. This approach is prohibitively costly since each network would have to be loaded with stations producing prescribed traffic and the number of stations required could be quite large, for example approaching 1000 to load the Ethernet fully.

The approach of a straightforward discrete-event simulation of a CSMA network with a large number of stations is also costly. To be realistic such a simulation would be required to model key events for each station including higher level functions, as well as access protocols.



General Structure of a Class of Contention-Type Local Computer Networks (Boxes are identified with the terminology used by Net One.)

The project, which is the subject of this report, was initiated by the U.S. Army Institute for Research in Management Information and Computer Science to develop a cost-effective alternative to the brute force or discrete-event simulation methods for studying CSMA networks. What is required is a means to assess the performance of particular contention networks for specific military applications at moderate cost.

The emulation facility being developed for studying the performance of CSMA bus networks is part of a more general facility--SPANET (System for Performance Analysis of Networks). SPANET is a dual-topology facility which will also support performance studies of mesh-type store-and-forward networks en completed.

Subsequent sections of the report describe the design, conce approach, and implementation, of the CSMA emulation facility and preliminary experiments performed to validate it. The appendix contains copies of published papers, one of which covers details of the development of the analytical model that the work is based on.

## II. DESIGN OF THE EMULATION FACILITY

The key innovation used in the emulation facility is the division of a potentially large number of network stations into a small group of Primary Stations and the remaining Background Stations. The Primary Stations are implemented in hardware, while the Background Stations are represented by artificial traffic generated with a computer program.

Figure 2.1 gives a logical view of the emulation facility. The two Primary Stations, A and B, operate essentially as if they are connected to a physical network. Actually they transfer data into and out of a shared micro-computer memory with the transfer being controlled with the combination of the Background Traffic Program, the Control Program and Switch shown in the figure. The Background Traffic Program generates a sequence of busy-idle intervals using an algorithm with the parameters of the Background Stations specified as input data.

Reduced to its simplest terms, the emulation facility can be represented as in Figure 2.2. The "Emulated Contention Network" consists of the common memory, the control program, the switch and additional circuits not shown in Figure 2.1. Implementation of this and other portions of the facility are discussed in later sections of the report. A more detailed discussion of the overall logical design of the facility is given in a paper by the project staff, [OREI 82 A], included in the report as Appendix A. The algorithm for producing the emulated background traffic and its verification are discussed briefly below. A detailed discussion of the same topics is given in a paper by O'Reilly and Hammond [OREI 82 B] included in the report as Appendix B.

The emulated background traffic is input to the emulated contention network as a binary function of time, B(t), delineating a random sequence of

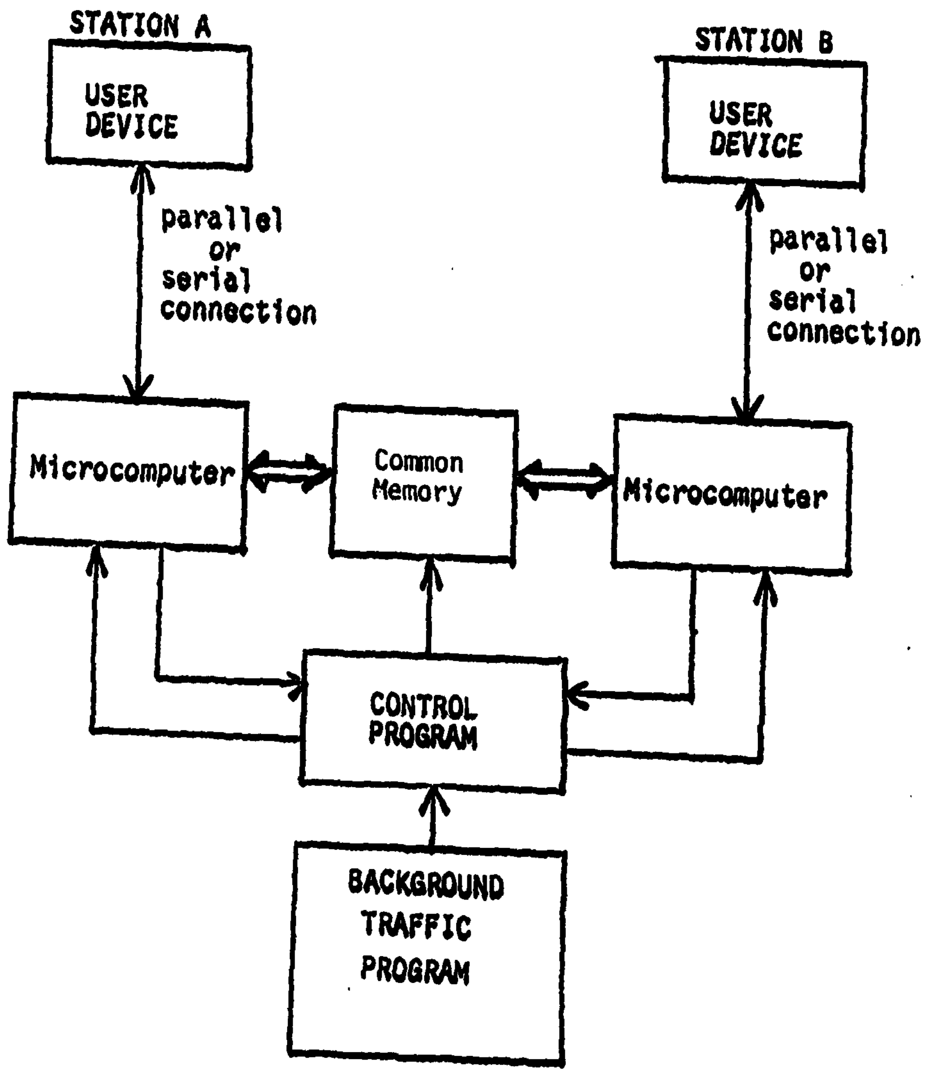


Figure 2.1 Logical Block Diagram of a Facility for Emulating Contention-Type Local Networks.

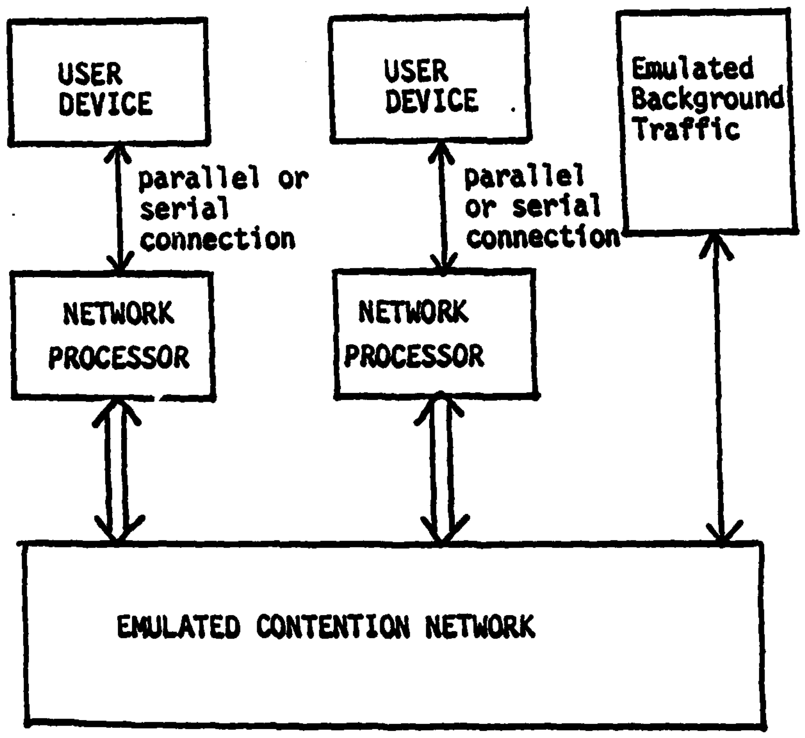

USER DEVICE
parallel or serial connection
NETWORK PROCESSOR
USER DEVICE
parallel or serial connection
NETWORK PROCESSOR
Emulated Background Traffic
EMULATED CONTENTION NETWORK

Figure 2.2 Block Diagram of the CSMA Emulation Facility

busy/idle periods as shown in Figure 2.3. A random sequence of times initiating and ending the busy periods is generated by an algorithm based on a set of equations describing the statistical nature of the background traffic. These equations are solved in a computer program which uses a modified Monte Carlo approach to generate the busy/idle intervals.

To achieve the required speed and flexibility, the traffic generation algorithm is run off-line and the random sequence of beginning and ending times for the busy periods is stored. Hardware in the emulated contention network of Figure 3 accesses the table of stored data in a manner which is synchronized to the stored time intervals.

An aspect of the operation, which received some attention in the design of the facility pertains to the interaction of the primary stations with the background traffic. A specific issue studied is the treatment of the background traffic during good transmissions by primary stations. In an actual network only one station can transmit good data at a time, so that the background stations must back-off during good transmission by primary stations. After studying two modes of accessing the background traffic, it was concluded that continuous sampling of the stored busy/idle intervals, but blanking this output during good transmissions by primary stations, gives better results than the alternative of deactivating the sampling of the stored intervals during good transmissions.

Derivation of the equations and logical structure of the algorithm for background traffic is given in the paper referenced above and in Appendix B. Required inputs are: the number of background stations, statistics of the bus lengths, propagation times, station loads, and protocols used, both specific access and some higher level. The number of stations and bus lengths can be changed easily. Arrivals to the stations are currently assumed to be Poisson

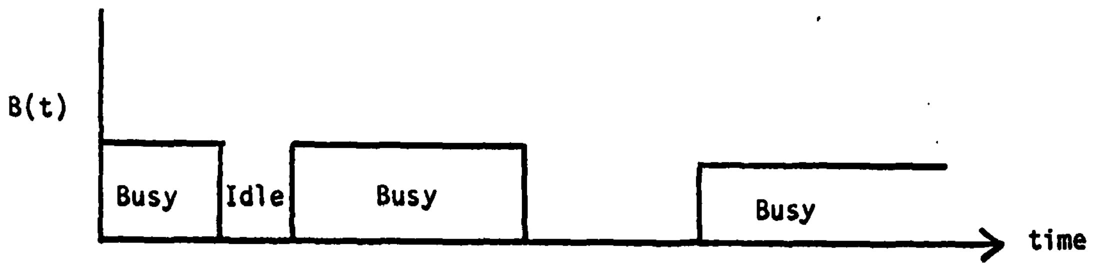


Figure 2.3 Output of the Traffic Generator

distributed, and this parameter could be difficult to change since it is incorporated into the equations of the algorithm. Protocols can be changed with routine changes to the algorithm.

The algorithm, programmed with the parameters of Ethernet, has been validated by comparison to an in-house discrete event simulation in studies of bus traffic. Excellent agreement is obtained for throughput over a range of 0 to 20 stations. Comparison is also made with measured results for Ethernet reported by Shoch and Hupp [SHOC 80]. Agreement in this case is also good, although not exact as would be expected in comparing theoretical and experimental data. Simulation studies of the interaction between primary and background stations have also been performed and good agreement with theory has been obtained.

A further contribution of the work and a verification of the mathematical model used for the algorithm is contained in a new closed-form expression for the throughput of a slotted 1-persistent CMSA/CD channel derived using the probabilistic equations of the algorithm and a number of simplifying assumptions. This equation, given in [OREI 82 B], reduces to well-known theoretical results for special cases.

The algorithm is implemented in FORTRAN code and runs on the PDP 11/70 at the AIRMICS facility. The program used for the background traffic is termed JNET. Other programs supplied are SIMNET and EXT. The three programs are described in a user's manual supplied under separate cover.

## III. HARDWARE IMPLEMENTATION

This section describes the hardware implementation of the bus emulation system. As described in the previous sections, the bus emulator approach is to model explicitly two nodes in the bus network with their user hosts and then model all the remaining modes on the network as a single statistical bus load. The bus traffic is observed through time delays by the two stations under study as they attempt to transmit messages to each other during the idle slots of the bus traffic. The following elements are required to implement this type of procedure.

1. Generation of an array of numbers to represent the busy/idle periods of the background bus traffic.

2. Hardware to convert the busy/idle data into logic levels as a function of time.

3. Hardware to implement programmable time delays to represent the physical delays produced by the length of the bus.

3. Transceiver nodes for the 2 stations under study that will implement the CSMA/CD network algorithm.

5. A method for generation of a data array to represent the statistical traffic between the 2 stations, or actual traffic from live hosts.

6. Hardware to take the array data and generate physical message traffic to the transceivers, when actual traffic is not used.

A hardware block diagram of the system used to implement the 6 elements is shown in Figure 3.1. In this system the master Nova 4 computer with its disk system, line printer and terminal, perform the master control of the bus emulation system. The statistical bus traffic information is generated off-line on a PDP 11/70 computer using the algorithms developed in the previous

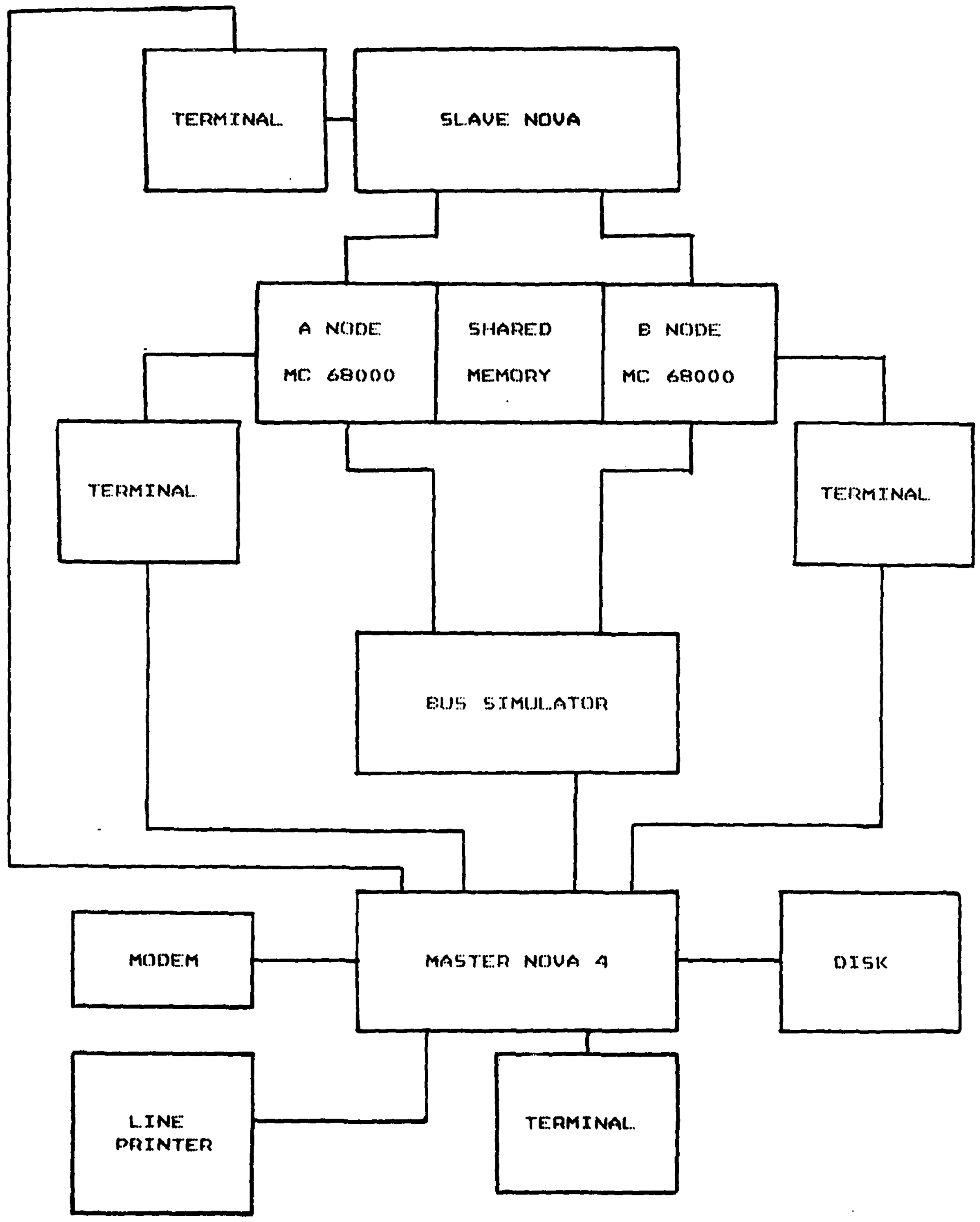


FIGURE 3.1 BUS NETWORK HARDWARE BLOCK DIAGRAM

sections. Generation of the busy/idle logic levels is performed by the bus simulator hardware under control of the master Nova 4. The time delays are also performed by the bus simulator hardware and are initialized by the master Nova 4 computer. The transceiver node implementation is performed by three 68000 Motorola microprocessor units with a common shared memory. Since the microprocessors can share memory locations, messages can be moved from one transceiver node to another extremely quickly. This savings in time allows the transceiver to implement most of the CSMA/CD network protocol in software rather than hardware. The statistical data used to determine the traffic between the two transceiver nodes is generated in software on the master Nova 4 computer and transferred to the slave Nova 4. The slave Nova 4 uses this data to generate the physical traffic between the two stations under study when artificial traffic is desired. When a particular bus traffic experiment has been completed, the two transceiver nodes transmit data back to the master Nova computer indicating the arrival time, departure time and number of attempts for each packet transmitted during the experiment. The Nova 4 computer analyzes this data and prints a summary of the experimental data on the system line printer.

The following sections are a detailed description of each of the 6 elements of the bus emulator.

**Generation of Bus Traffic Busy/Idle Patterns**

The busy/idle patterns used to represent the bus traffic supplied by the remaining stations on the bus are generated in software on a PDP 11/70 computer using the algorithms developed in the previous sections. By changing various parameters in the traffic generation program, a large variety of bus networks can be simulated. The various parameters that can be changed in this program are given in the Table 3.1 with their minimum and maximum values.

EXPERIMENT INPUTS

| PARAMETER | MIN. | MAX. |
|---|---|---|
| BUS TRAFFIC: | | |
| BUS RATE (MEGA BITS/SEC.) | .1 | 10.0 |
| BUS LENGTH (METERS) | 50 | 100000 |
| NUMBER OF STATIONS | 1 | 2000 |
| MESSAGE LENGTH (BITS) | 100 | 100000 |
| MEAN INTERARRIVAL TIME (MICRO SEC.) | X | 100000 |
| BACK OFF ALGORITHM | | |
| LINEAR INCREMENTAL | | |
| FIXED MEAN | | |
| BINARY EXPONENTAL | | |
| ACKNOWLEDGE LENGTH (BITS) | 0 | 100 |

TABLE 3.1 MAXIMUM AND MINIMUM BUS TRAFFIC PARAMETERS

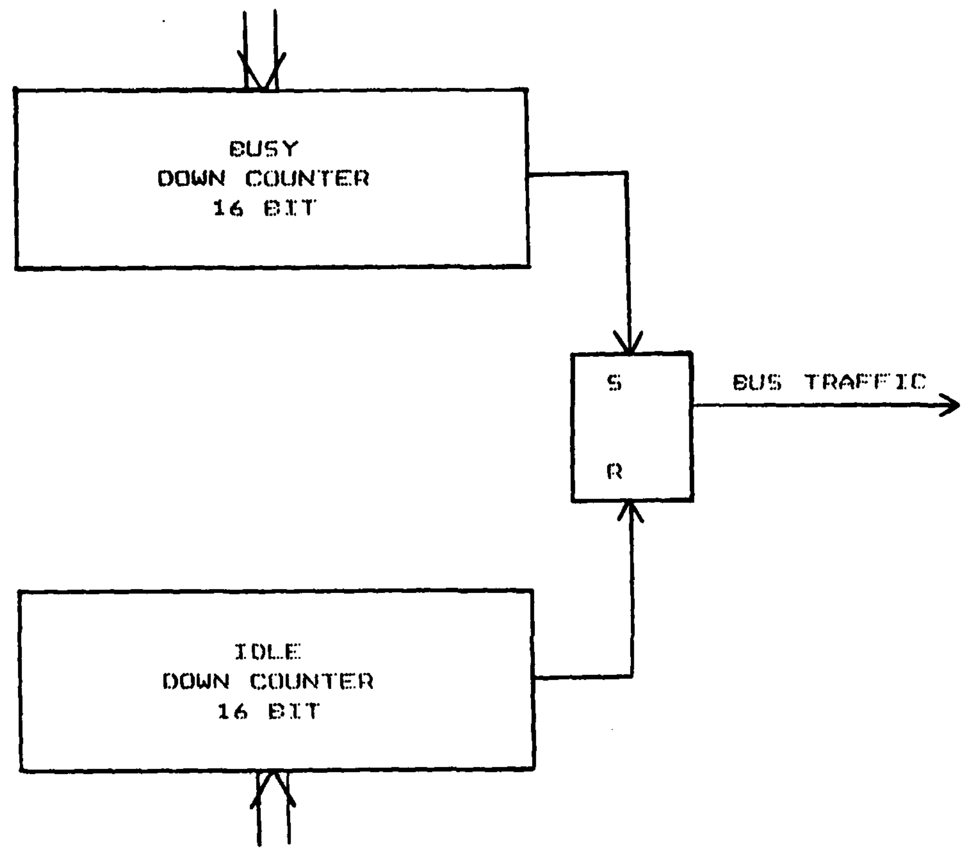
The output from this program is a sequence of numbers that represent in microseconds the consecutive busy/idle periods that would be present on the bus for the particular configuration simulated. These patterns are transferred from the PDP 11/70 to the Nova 4 master computer to simulate the bus traffic.

**Bus Traffic Logic Level Generation**

In order to use the busy/idle patterns generated by the PDP 11/70, these patterns must be converted to on/off logic levels as a function of real time. This conversion is accomplished by means of a bus simulation circuit interfaced to the master Nova 4 computer. A simplified block diagram of the bus simulator is shown in Figure 3.2. The circuit utilizes two 16-bit down counters to set or reset a flip-flop that produces the logic levels. At the beginning of the process the computer sets the first busy period into the busy counter and the first idle period into the idle counter. When the busy counter reaches zero, it resets the logic flip-flop to zero. The idle counter is then enabled and starts counting down to set the flip-flop back to 1. During this idle period the computer loads the next busy pattern into the busy counter. During the next busy period, the computer reloads the idle counter. This scheme continues until all of the patterns have been processed. When the process terminates the bus is held in the busy state to inhibit further transmission on the bus.

**Bus Time Delay Implementation**

The bus emulator implements all of the real time delays that would exist between the 2 stations and either of the stations and the bus simulator. A simplified block diagram of the time delay relationships are shown in Figure 3.3. The busy/idle patterns generated by the bus simulator described in the



MINIMUM BUSY/IDLE PERIOD = 6 MICRO SECONDS
MAXIMUM BUSY/IDLE PERIOD = 65 MILLI SECONDS

FIGURE 3.2 BUSY/IDLE PATTERN GENERATOR

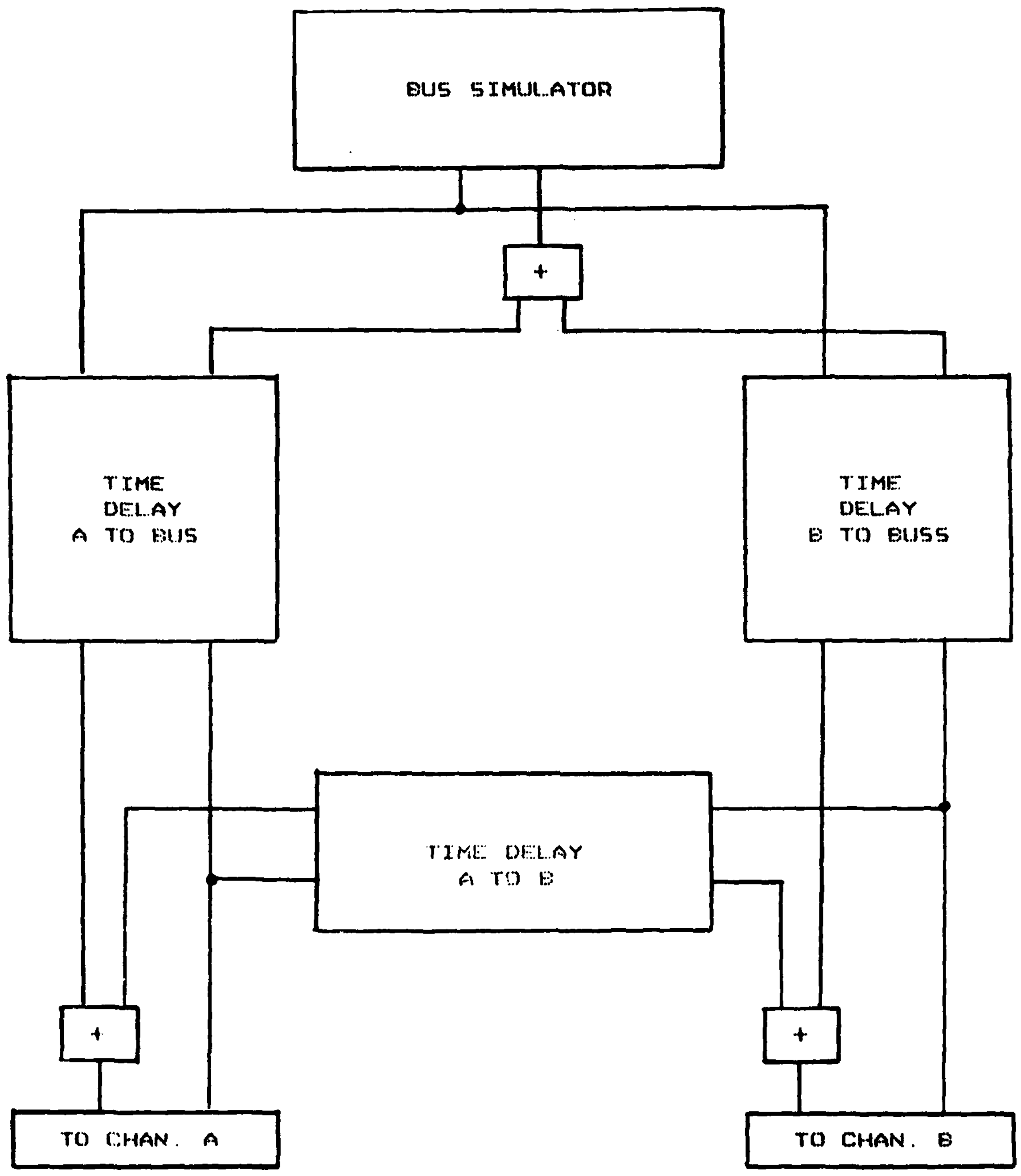


FIGURE 3.3 BUS SIMULATOR BLOCK DIAGRAM

previous sections is fed through a time delay to the A channel node and through a separate time delay to the B channel node. The transmit output from the A and B channel nodes are also fed through time delays back to the bus simulator. The delayed A and B channel transmit signals are read together and used to inhibit the bus simulator output when A or B transmit signals are present. The transmit signal from A is also fed through a time delay to the B node, and the B transmit signal is fed thro·gh a delay back to the A node. Each of the 6 time delay circuits is programmable from .2 to 409.5 microseconds from the master Nova 4 computer. Since the time delay between any 2 points in the network is the same in each direction, the 6 time delays are programmed as 3 pairs.

In order to implement time delays that could exceed the busy/idle pattern times, the time delay circuit must store a section of the bus signal and then output it a programmable delay time after it is read in. This is accomplished by use of a 1 bit by 4,096 RAM memory chip as shown in Figure 3.4. Every .1 microseconds the RAM memory is switched between a read cycle and a write cycle. The address during read cycle is determined by a 12 bit read counter and the address during the write cycle is deterined by a 12 bit write counter. These counters are also incremented every .1 microseconds. This means that alternately the data is being read into and out of different sections of the RAM memory. The difference between the read and write address will then determine the amount of delay between data being read into the RAM and data being read out of the RAM. This delay is programmed by allowing the Nova 4 to initially set the counter states. The read counter is always initialized to zero and the write counter is initially set to some value n. The read counter must then increment to a value of n before it reads the first data bit written. The net time delay is then n x clock period of .1 microsec-

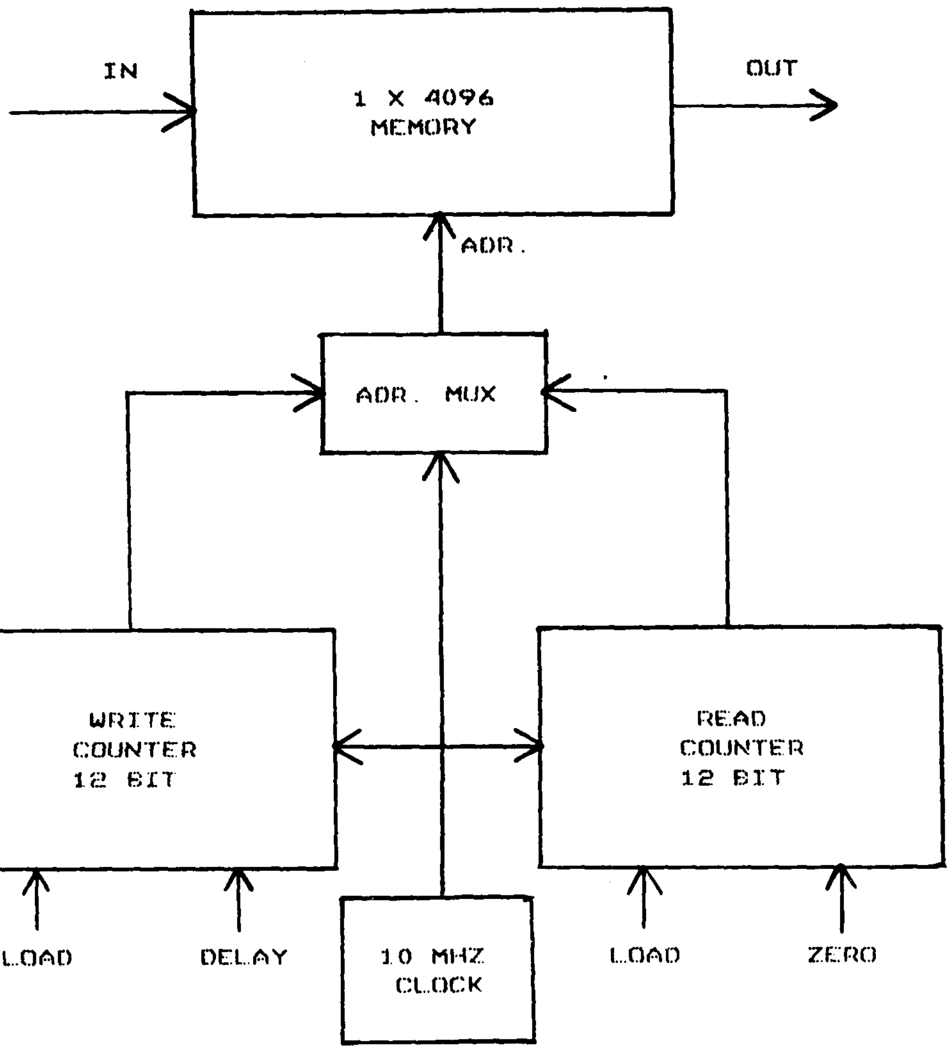


FIGURE 3.4 TIME DELAY HARDWARE BLOCK DIAGRAM

onds. Since the counters and the memory length are equal they will continue to cycle through the memory space with one initial setting from the computer.

## Implementation of Transceiver Nodes

The transceiver nodes used to emulate the 2 nodes on the bus that will be studied are implemented using 68000 microprocessor systems. The 2 node pair is implemented in 3 microprocessor systems sharing a common memory. Two of the microprocessors are used to implement the CSMA/CD protocols at the 2 nodes under test. The third microprocessor is used to receive serial messages from the slave Nova 4 and build message queues for both of the node microprocessors. Both the message queue and the current message being transmitted are passed from one microprocessor to the next through the shared memory. A block diagram of this configuration is shown in Figure 3-5 and the flow diagram of the CSMA/CD protocol is shown in Figure 3-6.

The following procedure is used to transmit a message through the system from one node to another.

1. A message is received by the 68000 IO processor either from the Nova 4 serial traffic generator or a live host, depending on the mode of operation. In an experimental mode the message content is unimportant, and the protocol for the serial messages is a simple 2 byte transmission indicating the number of characters of the message to be transmitted. Since long messages can be represented by a 2 byte transmission, this technique allows the 9600 baud serial link between the Nova 4 and the MC 68000 IO processor to simulate message transmissions at a much higher baud rate. When the system is used in a live host environment the protocol is changed to an actual message receiving and transmission protocol (Motorola defined), but the baud rate is limited to a maximum of 9600 baud.

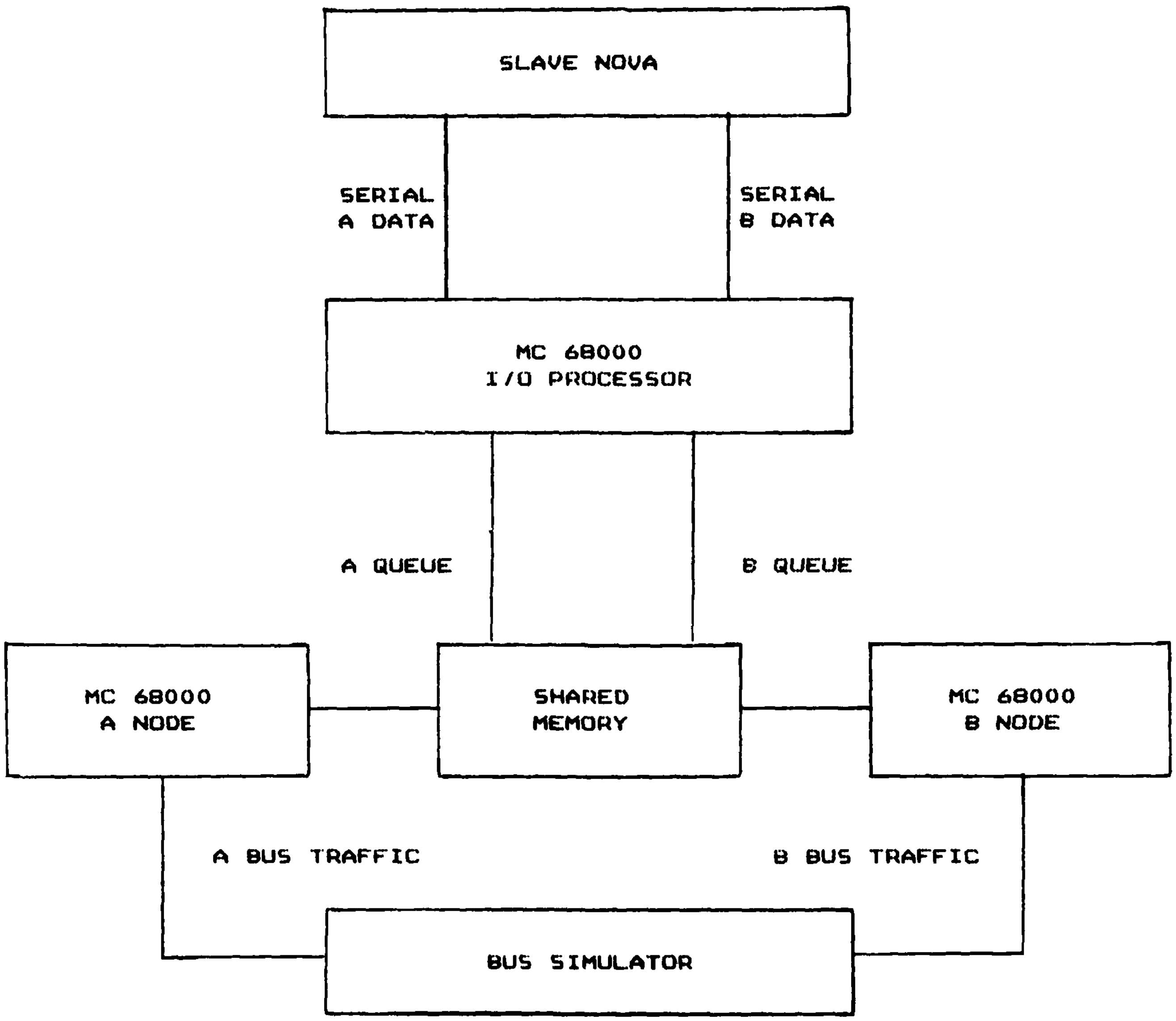


FIGURE 3.5 MC 68000 CONFIGURATION

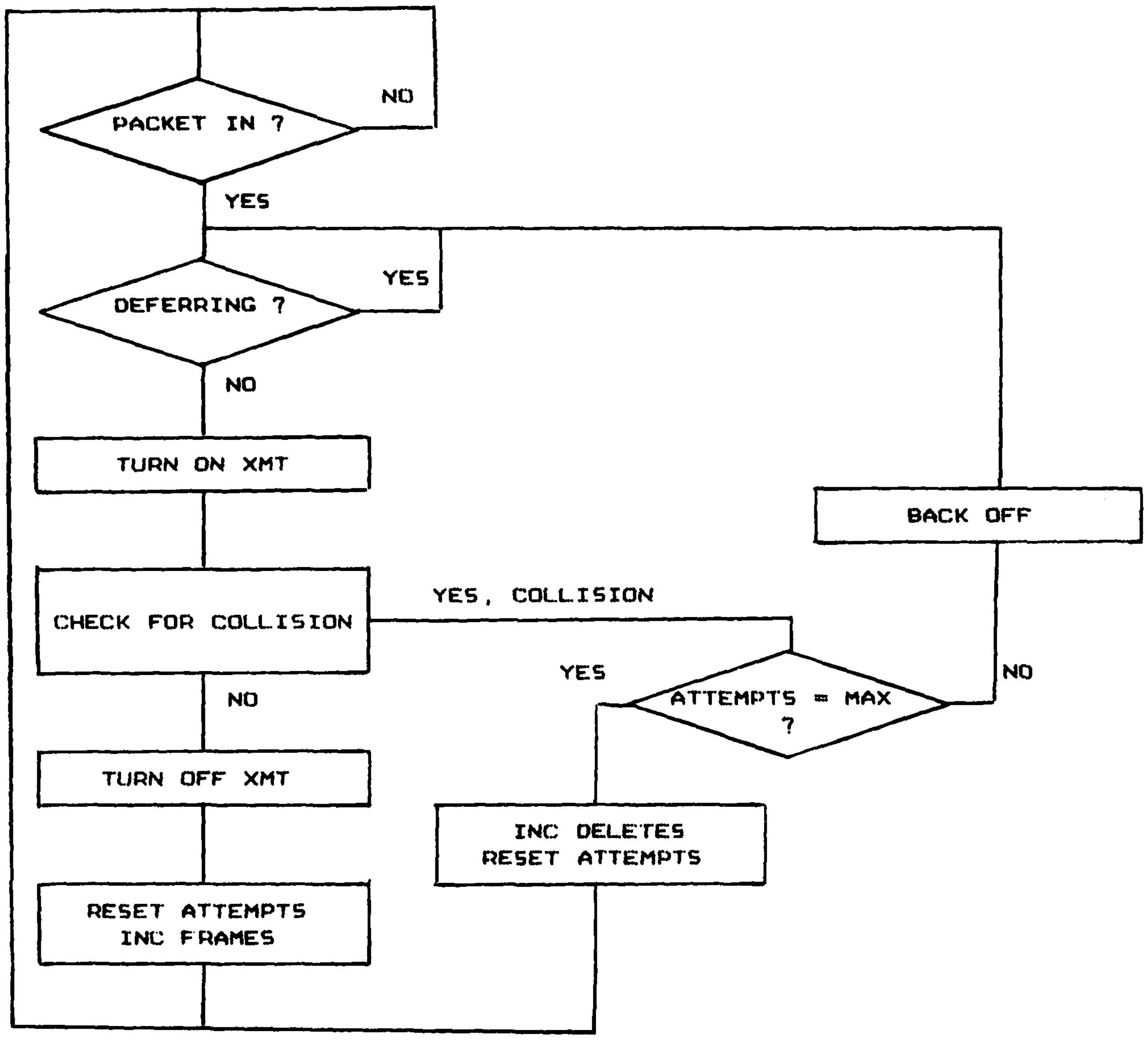


FIGURE 3.6 NODE SIMPLIFIED SOFTWARE FLOW CHART

2. The IO processor takes the messages for A and B and builds queues in the shared memory indicating the message number and the message length.

3. The A and B processors, which are programmed to implement the CSMA/CD protocol shown in Figure 3-6 will see the messages in the queue as a message ready.

4. The A or B processor would take the message length and number from the queue and read the time from the real time clock and store it as the starting time of the message.

5. The node processor will look at the busy/idle patterns from the bus simulator through a parallel IO port to determine when a message can be sent. As long as the bus is in a busy condition, the node processor remains in the deferred state and holds the message.

6. After 2 consecutive observations of the bus being in the idle condition, the node processor will start its message transmission by bringing the parallel IO output transmit line high.

7. The transmit line from the node processor and the bus signal from the bus simulator are and-ed together to set a collision flip-flop. Whenever the bus simulator and the transmit signals are in the busy state simultaneously, a collision has occurred. This flip flop can be read and reset by the node processor through the parallel IO board.

8. The node processor enters a timing loop that holds the transmit line high and observes the collision detector for a length of time that would normally be required for transmitting a message of the specified length.

9. If no collisions occur during the message transmission, the transmit line is returned to the low condition to terminate the transmission. The time is read from the real time clock and stored as the message ending time. Successful transmission of the message is indicated through the shared memory by removing the message from the message queue. The processor then returns to the start of the loop and awaits another message to be queued by the IO processor.
10. If a collision occurs the transmit line from the node processor is immediately returned low. The number of attempts is incremented and the current number of attempts is compared to a maximum value. If the number of attempts has exceeded the maximum then the message is deleted and the message delete count is incremented.
11. If the number of attempts does not exceed the maximum, then a back-off time is calculated using a random number generator. All the initial experiments using the experimental mode used a binary exponential back-off algorithm where the mean time for any back-off doubled as the number of attempts increased. When the number of attempts exceeded 10 the mean time per back-off remained constant.
12. When the back-off time expires the processor returns to the beginning of the loop and tries to retransmit the message.

The implementation of the CSMA/CD protocol in the 68000 microprocessor was tailored in particular to the binary exponential back-off algorithm. If other forms of back-off algorithm are to be implemented by the 68000 system, such as a fixed or linear back-off algorithm, then the particular section of assembly language code in the 68000 microprocessor would have to be replaced with an equivalent section for the particular algorithm chosen.

## Generation of Statistical Serial Data for Experimental Mode

The serial data for the bus experiment is generated by a separate software program on the Master Nova 4 computer. This program is run prior to the actual bus experiment and generates a disk file on the master disk indicating the length of each message and the time between messages. The program also prints a record on the system line printer indicating the file name of the data record and the type of statistics used to generate it. A large number of files can be generated based on varying statistical values so that they will be ready for use during bus experiments. A typical dialogue of the operations of the serial data generator is shown in Table 3-2. The minimum and maximum values of the input parameters are shown in Table 3-3 and a sample printout from the program operation is shown in Table 3-4.

## Hardware Implementation of the Serial Traffic

The serial traffic for the bus experiment is generated by the slave Nova 4 using two RS232 interfaces and an assembly language program that takes message information stored in memory and generates the 2 byte message transmission to the Motorola 68000 IO processor. The message data is stored in 2 separate memory buffers, one for messages to be sent to the A node processor and the other to be sent to node B processor. Both of the data buffers and the assembly language program are downloaded from the master Nova 4 computer. Startup and termination of the assembly language process is also controlled during bus experiments by the master Nova 4 computer. This means that the bus experiment operator needs only to interact with the master Nova 4 computer through its main console. Table 3-5 shows a typical program download operation and Table 3-6 shows a typical download operation for both the A and B data bases.

```
SDBG1<>

  STATISTICAL DATA BASE GENERATION PROGRAM - REV. 1.0

  ENTER MEAN MESSAGE LENGTH (BYTES) 100<>

  ENTER MESSAGE VARIATION (BYTES) 25<>

  ENTER MEAN INTERMESSAGE INTERVAL (MILLI SEC) 50<>

  ENTER INTERMESSAGE VARIATION (MILLI SEC) 10<>

  ENTER NUMBER OF MESSAGES TO GENERATE 200<>

  ENTER BAUDRATE OF SERIAL PORT 9600<>

  ENTER SPEED OF BUS (MEGABITS/SEC ) 1.4<>

  ENTER DATA BASE FILENAME SDB8.DF<>

  EQUIVALENT MESSAGES/SECOND = 6.486

  STOP

  12:05:15

  R
```

NOTE: OPERATOR RESPONSE UNDERLINED

TABLE 3.2 SERIAL DATA GENERATION DIALOG

EXPERIMENT INPUTS

| PARAMETER | MIN. | MAX. |
|---|---|---|
| SERIAL (USER) TRAFFIC: | | |
| MEAN MESSAGE LENGTH (BYTES) | 5 | 10000 |
| LENGTH VARIATION (BYTES +/-) | 0 | ML - 5 |
| INTERMESSAGE INTERVAL (MILLI SEC.) | 1 | 10000 |
| INTERVAL VARIATION (MILLI SEC.) | 0 | INT - 1 |
| NUMBER OF MESSAGES | 1 | 5000 |
| SERIAL PORT BAUDRATE | 110 | 50000 |

TABLE 3.3 MAXIMUM AND MINIMUM SERIAL TRAFFIC PARAMETERS

```
     STATISTICAL DATA BASE GENERATION PROGRAM - REV 1.0
******************************************
     DATA BASE FILENAME = SDB8.DF
******************************************

MEAN MESSAGE LENGTH (BYTES) =    100
MEAN MESSAGE VARIATION (BYTES) =     25
MEAN INTERMESSAGE INTERVAL (MILLI SEC.) =     50
MEAN INTERMESSAGE VARIATION (MILLI SEC.) =     10
NUMBER OF MESSAGES GENERATED =    200
SERIAL PORT BAUD RATE =   9600
EQUIVALENT MESSAGES/SECOND. =   6.486
```

TABLE 3.4 SERIAL DATA GENERATOR PRINT OUT

```
NSLDR<>
      NOVA SLAVE LOADER - REV. 1.0
      ENTER LOAD FILENAME STFG5.LS<>
   [ NUMBERS WILL APPEAR HERE ]
   STOP
   12:05:15
   R
NOTE: OPERATORS RESPONSE UNDERLINED
```

TABLE 3.5 NOVA SLAVE PROGRAM DOWNLOAD PROCEDURE

```
DBLDR<>
      NOVA SLAVE DATA BASE LOADER - REV. 1.0
      ENTER A CHANNEL DATA BASE (OR NONE) SDB3.DF<>
      ENTER B CHANNEL DATA BASE (OR NONE) SDB2.DF<>
   STOP
   12:05:15
   R
NOTE: OPERATOR RESPONSE UNDERLINED
```

TABLE 3.6 SLAVE NOVA DATA DOWNLOAD PROCEDURE

## IV. BUS EXPERIMENT

In the previous section, each component of the network system is described. In this section the procedure for setting up these component parts into a complete bus experiment will be discussed. A flow chart showing the hardware and software components in the interaction of setting up a bus experiment is illustrated in Figure 4-1. This procedure is outlined as follows:

1. The master Nova 4 computer is linked to the AIRMICS PDP 11/70 through a data link program (PDP 11.SV). Through this program the background traffic simulation program (JNET) is executed and the data from the simulation is captured by the master Nova and stored as a disk file (BSIM.DF) on the system disk. A large number of these files can be generated with varying statistical characteristics for later use in bus traffic experiments.
2. Serial data is obtained by using the serial traffic generation program (SDBG1.SV). The serial data is also stored as disk files (SDB.DF) on the system disk. A large number of serial data files may also be generated in advance and called up for different bus experiments.
3. The slave Nova serial traffic generation program (STFG4.LS) is down loaded from the master Nova using the Nova slave download program (NSLDR.SB). The serial traffic data is also downloaded from the master Nova 4 computer using the data base downloader (DBLBR.SB).
4. After all network cables are checked for proper configuration and the 68000 microprocessors have been initialized, the bus experiment is started by executing the bus experiment control program (BSIM6.SV). This program controls the complete operation of the bus

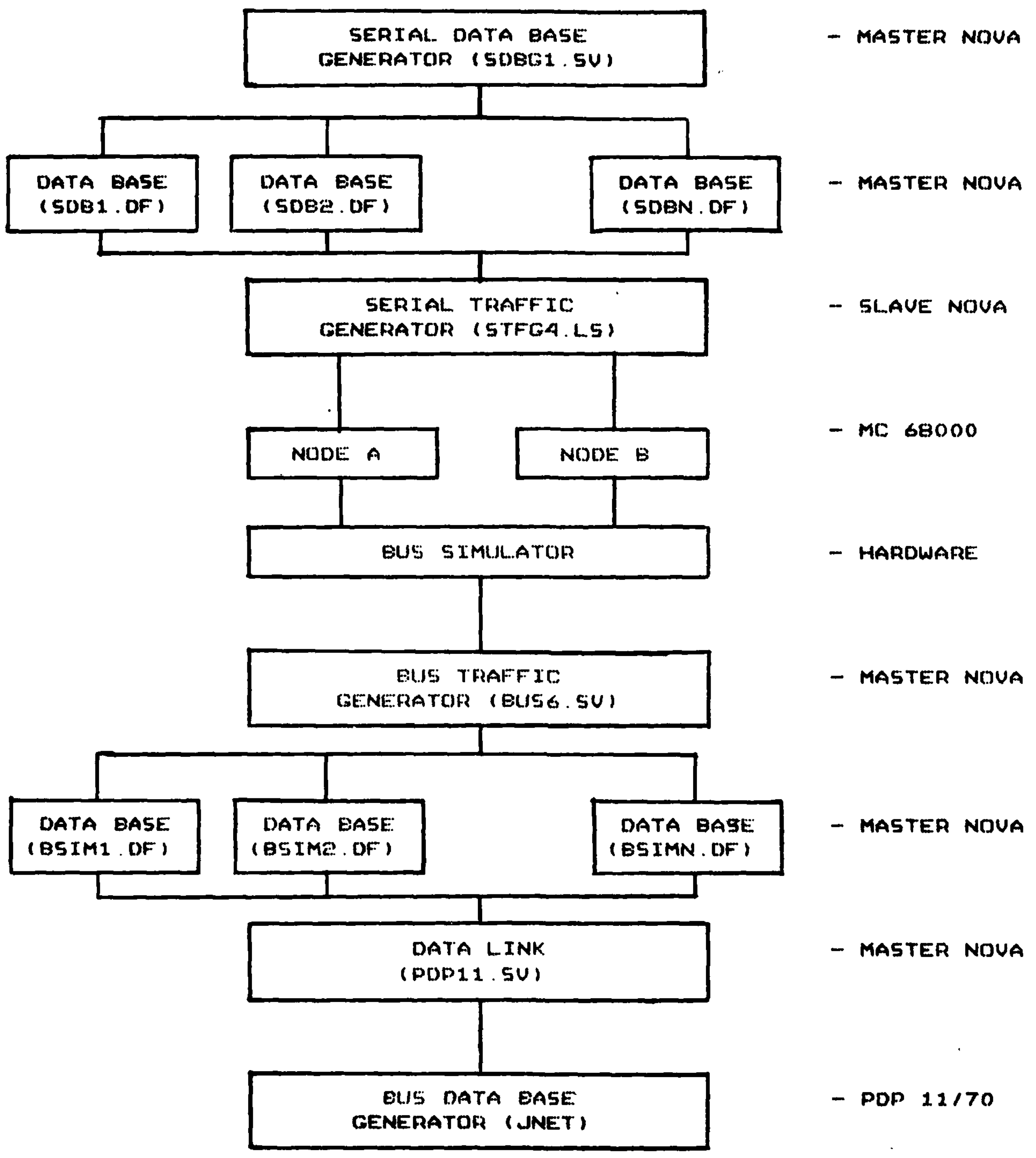


FIGURE 4.1 SOFTWARE FLOW DIAGRAM

experiment and collects data at the end of the experiment on the starting time, ending time and number of attempts of each message that is transmitted during the experiment. A typical main console dialogue of an experiment is given in Table 4-1. At the end of the experiment a complete statistical breakdown of the bus experiment is printed on the system line printer for both channel A and channel B. Included in this printout is the experimental setup message completions, cancellations, and back-offs. Time delay information and a histogram of number of message and time delay versus attempts. A typical printout is shown in Table 4-2. Upon request the system will also printout a description of each individual message that occured in an experiment including the length of the message, the number of attempts the message required and the delay time in processing the message. A typical printout is shown in Table 4-3.

```
BUS6<>

  GA. TECH/AIRMICS COMPUTER NETWORK

  BUS SIMULATION PROGRAM            - REV. 1.0

  ENTER SIMULATION DATA FILE NAME BSIM6.DF<>

       BUS TIME DELAYS - MIN.=.2, MAX.=409.4

  ENTER DELAY BETWEEN SIMULATOR AND CHAN A 1<>

  ENTER DELAY BETWEEN SIMULATOR AND CHAN B 2<>

  ENTER DELAY BETWEEN CHAN A AND CHAN B 3<>

  ENTER EXPERIMENT RUN TIME IN M.S. 5000<>


  NO. OF PATTERNS REQUIRED FOR RUNTIME = 2992


  NUMBER OF PATTERNS REQUIRED EXCEEDS

  NUMBER OF PATTERNS IN DATA FILE

       NUMBER REQUIRED = 2992

       NUMBER IN DATA FILE =  122

  POSSIBLE OPTIONS:

       1 - REPEAT DATA TO FORM  2992 PATTERNS

       2 - REDUCE REQUIRED TO    122 PATERNS

       3 - RUN CONTINUOUS PATTERNS

       4 - ABORT

  OPTION 1<>

  BUS SIMULATOR READY TO RUN 2992 PATERNS

  ENTER G TO START SIMULATION G<>
```

TABLE 4.1 BUS EXPERIMENT DIALOG

```
BUS SIMULATION RUN COMPLETE WITH 3063 PATTERNS

NO. OF SERIAL MESSAGES SENT TO A CHAN = 98

NO. OF SERIAL MESSAGES SENT TO B CHAN = 98

READ DATA FROM STATION A (Y OR N) ? Y<>
                                    ---
PRINT INDIVIDUAL MESSAGE DATA (Y OR N) ? Y<>
                                         ---
READ DATA FROM STATION B (Y OR N) ? Y<>
                                    ---
PRINT INDIVIDUAL MESSAGE DATA (Y OR N) ? Y<>
                                         ---

DO YOU WANT TO MAKE ANOTHER RUN (Y OR N) N<>
                                         ---
STOP

12:05:15

R
```

TABLE 4.1 (CONT.) BUS EXPERIMENT DIALOG

```
GEORGIA TECH - AIRMICS BUS SIMULATION NETWORK

      STATISTICAL DATA


      *** CHANNEL A ***


BUS SIMULATION FILE                  = BSIM6.DF
EXPERIMENT RUN TIME                  =  5000. MILLISEC.
NUMBER OF BUSY/IDLE PATTERNS         =  2992
DELAY - SIMULATOR TO CHANNEL A       =    1.0 MICROSEC.
DELAY - SIMULATOR TO CHANNEL B       =    2.0 MICROSEC.
DELAY - CHANNEL A TO CHANNEL B       =    3.0 MICROSEC.

NO. OF MESSAGES SENT TO CHANNEL   =    98
NO. OF MESSAGES COMPLETED         =    98
NO. OF MESSAGES CANCELED          =     0
NO. OF MESSAGES LEFT IN CHANNEL   =     0
TOTAL NO. OF BACKOFFS             =   190

MAXIMUM TIME DELAY =  77.328 MILLISEC.
MINIMUM TIME DELAY =   1.008 MILLISEC.
MEAN TIME DELAY    =   4.494 MILLISEC.

TOTAL BYTES PROCESSED      =     9800.
MAXIMUM MESSAGE LENGTH     =       100 BYTES
MINIMUM MESSGAE LENGTH     =       100 BYTES
EQUIVALENT MESSAGES/SEC.  =   19.600
MIN. EQUIVALENT BAUDRATE  =    19600.
EQUIVALENT UNLOADED BUS   =   222496. BPS


MESSAGE DISTRIBUTION

ATTEMPTS   NUMBER   DELAY TIME

    1        11        1.653
    2        53        2.133
    3        13        3.988
    4         6        5.114
    5         7        4.785
    6         2        6.389
    7         1        5.942
    8         1       12.403
    9         0        0.000
   10         2       18.787
   11         0        0.000
   12         1       47.184
   13         0        0.000
   14         1       77.328
   15         0        0.000
   16         0        0.000
```

TABLE 4.2 SAMPLE BUS EXPERIMENT PRINT OUTPUT

```
GEORGIA TECH - AIRMICS BUS SIMULATION NETWORK

INDIVIDUAL MESSAGE DATA

      MESSAGE  LENGTH   ATTEMPTS  DELAY TIME
           1      100        2        1.229
           2      100        3        4.752
           3      100       14       77.328
           4      100        1        1.872
           5      100        2        1.123
           6      100        2        1.114
           7      100        7        5.942
           8      100        2        1.123
           9      100        2        5.117
          10      100        1        3.955
          11      100        3        4.061
          12      100        1        1.018
          13      100        2        3.802
          14      100        2        1.402
          15      100        3        5.501
          16      100        2        1.718
          17      100        3        4.013
          18      100        2        1.930
          19      100       12       47.184
          20      100        1        1.603
          21      100        5        6.509
          22      100        2        1.114
          23      100        4        6.221
          24      100        2        1.411
          25      100        1        1.008
          26      100        2        1.440
          27      100        2        2.333
          28      100        2        1.987
          29      100        2        2.928
          30      100        1        1.680
          31      100        3        5.616
          32      100        4        4.061
          33      100        2        1.114
          34      100        2        1.142
          35      100        4        3.014
          36      100        2        1.114
          37      100        2        3.264
          38      100        3        3.763
          39      100        2        2.074
          40      100        2        1.584
          41      100        2        1.123
          42      100        4        5.779
          43      100        2        1.670
          44      100        2        2.813
          45      100        5        3.168
          46      100        1        1.181
          47      100        2        3.677
          48      100        2        4.205
          49      100        4        5.395
          50      100        2        1.123
          51      100        6        6.653
          52      100        4        6.211
          53      100        2        3.360
          54      100        3        2.813
```

| | | | |
|---|---|---|---|
| 55 | 100 | 2 | 1.824 |
| 56 | 100 | 5 | 5.846 |
| 57 | 100 | 3 | 2.390 |
| 58 | 100 | 6 | 6.125 |
| 59 | 100 | 2 | 1.123 |
| 60 | 100 | 5 | 3.965 |
| 61 | 100 | 2 | 2.486 |
| 62 | 100 | 2 | 1.853 |
| 63 | 100 | 2 | 7.018 |
| 64 | 100 | 2 | 1.123 |
| 65 | 100 | 2 | 3.043 |
| 66 | 100 | 10 | 12.998 |
| 67 | 100 | 2 | 1.306 |
| 68 | 100 | 2 | 1.114 |
| 69 | 100 | 1 | 1.392 |
| 70 | 100 | 5 | 3.235 |
| 71 | 100 | 1 | 2.054 |
| 72 | 100 | 2 | 1.344 |
| 73 | 100 | 3 | 1.248 |
| 74 | 100 | 5 | 5.338 |
| 75 | 100 | 2 | 2.170 |
| 76 | 100 | 3 | 5.088 |
| 77 | 100 | 2 | 1.459 |
| 78 | 100 | 3 | 6.758 |
| 79 | 100 | 2 | 2.995 |
| 80 | 100 | 1 | 1.402 |
| 81 | 100 | 2 | 3.206 |
| 82 | 100 | 10 | 24.576 |
| 83 | 100 | 3 | 2.870 |
| 84 | 100 | 8 | 12.403 |
| 85 | 100 | 2 | 1.517 |
| 86 | 100 | 1 | 1.018 |
| 87 | 100 | 2 | 2.717 |
| 88 | 100 | 2 | 2.237 |
| 89 | 100 | 2 | 2.698 |
| 90 | 100 | 2 | 3.619 |
| 91 | 100 | 2 | 1.853 |
| 92 | 100 | 2 | 1.891 |
| 93 | 100 | 2 | 1.728 |
| 94 | 100 | 3 | 2.966 |
| 95 | 100 | 5 | 5.434 |
| 96 | 100 | 2 | 1.114 |
| 97 | 100 | 2 | 1.267 |
| 98 | 100 | 2 | 2.314 |

TABLE 4.3 INDIVIDUAL MESSAGE PRINT OUT

# V. EMULATOR VERIFICATION

To verify the proper operation of the bus emulator, certain benchmark experiments were run under known experimental conditions. Since the bus experiments are very statistical in nature the extreme loading conditions were chosen to force the system into a deterministic behavior. Tests were run at no bus loading, full bus loading and constant collision mode of bus traffic. Following these extreme conditions a typical bus experiment was run and compared to a computer simulation of the same type experiment.

## No Load Test

In the no load test, 99 messages of 100 bytes each were transmitted through channel A with a 1 megabyte bus equivalent. The background traffic and the channel B traffic were set to 0 so that channel A's messages would not differ or collide. Under these conditions the experiment should indicate all 99 messages completed with no back-offs and the time delay for each message should be 1 millisecond. A printout of the experiment is shown in Table 5-1 indicating that all messages were completed with a maximum time delay of 1.018 and a minimum time delay of 1.008. The printout of the individual messages in Table 5-2 indicates that each message was exactly 100 characters and passed with one attempt and approximately 1 millisecond delay.

## Full Load Experiment

In the full load experiment 98 messages were sent to channel A, but the background traffic generator was kept in the busy state during the entire experimental run time. Under these conditions, the channel A should remain in the defer mode trying to send the first message transmission without success. At the end of the experiment, all the messages should be left in the

```
GEORGIA TECH - AIRMICS BUS SIMULATION NETWORK

     STATISTICAL DATA


     *** CHANNEL A ***


BUS SIMULATION FILE                    = BSIM5.DF
EXPERIMENT RUN TIME                    =  5000. MILLISEC.
NUMBER OF BUSY/IDLE PATTERNS           =  3063
DELAY - SIMULATOR TO CHANNEL A         =   1.0 MICROSEC.
DELAY - SIMULATOR TO CHANNEL B         =   2.0 MICROSEC.
DELAY - CHANNEL A TO CHANNEL B         =   3.0 MICROSEC.

NO. OF MESSAGES SENT TO CHANNEL    =     99
NO. OF MESSAGES COMPLETED          =     99
NO. OF MESSAGES CANCELED           =      0
NO. OF MESSAGES LEFT IN CHANNEL    =      0
TOTAL NO. OF BACKOFFS              =      0

MAXIMUM TIME DELAY =    1.018 MILLISEC.
MINIMUM TIME DELAY =    1.008 MILLISEC.
MEAN TIME DELAY    =    1.014 MILLISEC.

TOTAL BYTES PROCESSED      =     9900.
MAXIMUM MESSAGE LENGTH     =       100 BYTES
MINIMUM MESSGAE LENGTH     =       100 BYTES
EQUIVALENT MESSAGES/SEC.  =   19.800
MIN. EQUIVALENT BAUDRATE  =    19800.
EQUIVALENT UNLOADED BUS   =   985715. BPS


MESSAGE DISTRIBUTION

ATTEMPTS  NUMBER  DELAY TIME

    1       99     1.014
    2        0     0.000
    3        0     0.000
    4        0     0.000
    5        0     0.000
    6        0     0.000
    7        0     0.000
    8        0     0.000
    9        0     0.000
   10        0     0.000
   11        0     0.000
   12        0     0.000
   13        0     0.000
   14        0     0.000
   15        0     0.000
   16        0     0.000
```

TABLE 5.1 NO LOAD BUS EXPERIMENT PRINT OUTPUT

```
GEORGIA TECH - AIRMICS BUS SIMULATION NETWORK

INDIVIDUAL MESSAGE DATA

      MESSAGE  LENGTH    ATTEMPTS  DELAY TIME
         1      100         1      1.008
         2      100         1      1.018
         3      100         1      1.018
         4      100         1      1.018
         5      100         1      1.018
         6      100         1      1.018
         7      100         1      1.018
         8      100         1      1.018
         9      100         1      1.018
        10      100         1      1.018
        11      100         1      1.018
        12      100         1      1.018
        13      100         1      1.008
        14      100         1      1.008
        15      100         1      1.018
        16      100         1      1.018
        17      100         1      1.008
        18      100         1      1.018
        19      100         1      1.018
        20      100         1      1.008
        21      100         1      1.018
        22      100         1      1.018
        23      100         1      1.018
        24      100         1      1.008
        25      100         1      1.018
        26      100         1      1.018
        27      100         1      1.018
        28      100         1      1.018
        29      100         1      1.018
        30      100         1      1.008
        31      100         1      1.018
        32      100         1      1.018
        33      100         1      1.008
        34      100         1      1.008
        35      100         1      1.018
        36      100         1      1.008
        37      100         1      1.008
        38      100         1      1.018
        39      100         1      1.008
        40      100         1      1.008
        41      100         1      1.018
        42      100         1      1.018
        43      100         1      1.008
        44      100         1      1.008
        45      100         1      1.018
        46      100         1      1.008
        47      100         1      1.018
        48      100         1      1.018
        49      100         1      1.008
        50      100         1      1.008
        51      100         1      1.018
        52      100         1      1.018
        53      100         1      1.018
        54      100         1      1.008
```

| | | | |
|---|---|---|---|
| 55 | 100 | 1 | 1.008 |
| 56 | 100 | 1 | 1.018 |
| 57 | 100 | 1 | 1.018 |
| 58 | 100 | 1 | 1.018 |
| 59 | 100 | 1 | 1.018 |
| 60 | 100 | 1 | 1.018 |
| 61 | 100 | 1 | 1.008 |
| 62 | 100 | 1 | 1.008 |
| 63 | 100 | 1 | 1.018 |
| 64 | 100 | 1 | 1.018 |
| 65 | 100 | 1 | 1.018 |
| 66 | 100 | 1 | 1.018 |
| 67 | 100 | 1 | 1.018 |
| 68 | 100 | 1 | 1.018 |
| 69 | 100 | 1 | 1.018 |
| 70 | 100 | 1 | 1.008 |
| 71 | 100 | 1 | 1.008 |
| 72 | 100 | 1 | 1.018 |
| 73 | 100 | 1 | 1.018 |
| 74 | 100 | 1 | 1.018 |
| 75 | 100 | 1 | 1.008 |
| 76 | 100 | 1 | 1.018 |
| 77 | 100 | 1 | 1.018 |
| 78 | 100 | 1 | 1.018 |
| 79 | 100 | 1 | 1.008 |
| 80 | 100 | 1 | 1.018 |
| 81 | 100 | 1 | 1.008 |
| 82 | 100 | 1 | 1.018 |
| 83 | 100 | 1 | 1.008 |
| 84 | 100 | 1 | 1.018 |
| 85 | 100 | 1 | 1.018 |
| 86 | 100 | 1 | 1.008 |
| 87 | 100 | 1 | 1.018 |
| 88 | 100 | 1 | 1.018 |
| 89 | 100 | 1 | 1.018 |
| 90 | 100 | 1 | 1.018 |
| 91 | 100 | 1 | 1.018 |
| 92 | 100 | 1 | 1.018 |
| 93 | 100 | 1 | 1.018 |
| 94 | 100 | 1 | 1.018 |
| 95 | 100 | 1 | 1.018 |
| 96 | 100 | 1 | 1.008 |
| 97 | 100 | 1 | 1.008 |
| 98 | 100 | 1 | 1.018 |
| 99 | 100 | 1 | 1.008 |

TABLE 5.2 INDIVIDUAL MESSAGE PRINT OUT

channel with no back-offs taking place. The printout from this experiment is shown in Table 5-3.

## Constant Back-Off Experiment

In this experiment the collision flip-flop for channel A is held in the high state so that every time channel A tries to send a message, it will get a collision indication. Under these conditions, the channel will try to process each message 16 times getting a collision each attempt. After 16 attempts the messages will be cancelled and a new message will be brought in. The printout from this experiment shown in Table 5-4 indicates that 98 messages were sent to channel A with no completions. The system cancelled 90 of the messages with 1,442 back-offs and left 8 messages unprocessed at the termination of the experiment.

## Comparison of Bus Emulator to Computer Simulation

A final verification was performed by comparing the bus emulator to a computer simulation of the same bus traffic scenario. A typical scenario outlined in Table 5-5 was setup in the computer simulation and on the bus emulator. The resulting data was collected and summarized in Table 5-6. Since these experiments are very statistical in nature, the data from the 2 systems were not expected to be identical. The number of messages completed and the number of back-offs were extremely close but the mean time delay of the simulation was approximately 1 milisecond longer than the emulator. It should be noted that a single message requiring a large number of attempts for completion can cause a significant change in the mean time delay. An example of this would be the message in the simulation that required 15 attempts and used 97.7 milisecond of time delay. The most significant difference in the comparison is the fact that in the simulation run, almost half of the messages

```
GEORGIA TECH - AIRMICS BUS SIMULATION NETWORK

      STATISTICAL DATA


      *** CHANNEL A ***

BUS SIMULATION FILE                  = BSIM6.DF
EXPERIMENT RUN TIME                  =  5000. MILLISEC.
NUMBER OF BUSY/IDLE PATTERNS         =  2992
DELAY - SIMULATOR TO CHANNEL A       =    1.0 MICROSEC.
DELAY - SIMULATOR TO CHANNEL B       =    2.0 MICROSEC.
DELAY - CHANNEL A TO CHANNEL B       =    3.0 MICROSEC.

NO. OF MESSAGES SENT TO CHANNEL   =    98
NO. OF MESSAGES COMPLETED          =     0
NO. OF MESSAGES CANCELED           =     0
NO. OF MESSAGES LEFT IN CHANNEL   =    98
TOTAL NO. OF BACKOFFS              =     0

MAXIMUM TIME DELAY =    0.000 MILLISEC.
MINIMUM TIME DELAY =    0.000 MILLISEC.
MEAN TIME DELAY    =    0.000 MILLISEC.

TOTAL BYTES PROCESSED      =        0.
MAXIMUM MESSAGE LENGTH     =        0 BYTES
MINIMUM MESSGAE LENGTH     =        0 BYTES
EQUIVALENT MESSAGES/SEC.   =        0
MIN. EQUIVALENT BAUDRATE   =        0.
EQUIVALENT UNLOADED BUS    =        0. BPS


MESSAGE DISTRIBUTION

ATTEMPTS   NUMBER   DELAY TIME

     1        0       0.000
     2        0       0.000
     3        0       0.000
     4        0       0.000
     5        0       0.000
     6        0       0.000
     7        0       0.000
     8        0       0.000
     9        0       0.000
    10        0       0.000
    11        0       0.000
    12        0       0.000
    13        0       0.000
    14        0       0.000
    15        0       0.000
    16        0       0.000
```

TABLE 5.3 NO LOAD BUS EXPERIMENT PRINT OUTPUT

```
GEORGIA TECH - AIRMICS BUS SIMULATION NETWORK

     STATISTICAL DATA


     *** CHANNEL A ***


BUS SIMULATION FILE                  = BSIM6.DF
EXPERIMENT RUN TIME                  =  5000. MILLISEC.
NUMBER OF BUSY/IDLE PATTERNS         =  2992
DELAY - SIMULATOR TO CHANNEL A       =   1.0 MICROSEC.
DELAY - SIMULATOR TO CHANNEL B       =   2.0 MICROSEC.
DELAY - CHANNEL A TO CHANNEL B       =   3.0 MICROSEC.

NO. OF MESSAGES SENT TO CHANNEL   =    98
NO. OF MESSAGES COMPLETED         =     0
NO. OF MESSAGES CANCELED          =    90
NO. OF MESSAGES LEFT IN CHANNEL   =     8
TOTAL NO. OF BACKOFFS             =  1442

MAXIMUM TIME DELAY =   0.000 MILLISEC.
MINIMUM TIME DELAY =   0.000 MILLISEC.
MEAN TIME DELAY    =   0.000 MILLISEC.

TOTAL BYTES PROCESSED     =        0.
MAXIMUM MESSAGE LENGTH    =        0 BYTES
MINIMUM MESSGAE LENGTH    =        0 BYTES
EQUIVALENT MESSAGES/SEC.  =        0
MIN. EQUIVALENT BAUDRATE  =        0.
EQUIVALENT UNLOADED BUS   =        0.  BPS


MESSAGE DISTRIBUTION

ATTEMPTS  NUMBER  DELAY TIME

    1        0      0.000
    2        0      0.000
    3        0      0.000
    4        0      0.000
    5        0      0.000
    6        0      0.000
    7        0      0.000
    8        0      0.000
    9        0      0.000
   10        0      0.000
   11        0      0.000
   12        0      0.000
   13        0      0.000
   14        0      0.000
   15        0      0.000
   16        0      0.000
```

TABLE 5.4 NO LOAD BUS EXPERIMENT PRINT OUTPUT

BENCKMARK EXPERIMENT - SIMULATOR/EMULATOR

EXPERIMENT INPUTS

| PARAMETER | VALUE |
|---|---|
| BUS TRAFFIC: | |
| BUS RATE (MEGA BITS/SEC.) | 1.0 |
| BUS LENGTH (METERS) | 1150 |
| NUMBER OF STATIONS | 40 |
| MESSAGE LENGTH (BITS) | 1000 |
| MEAN INTERARRIVAL TIME (MICRO SEC.) | 50000 |
| BACK OFF ALGORITHM | BIN. EXP. |
| ACKNOWLEDGE LENGTH (BITS) | 0 |
| SERIAL (USER) TRAFFIC: | |
| MEAN MESSAGE LENGTH (BYTES) | 100 |
| LENGTH VARIATION (BYTES +/-) | 0 |
| INTERMESSAGE INTERVAL (MILLI SEC.) | 10 |
| INTERVAL VARIATION (MILLI SEC.) | 0 |
| NUMBER OF MESSAGES | 100 |
| SERIAL PORT BAUDRATE | 25000 |
| EXPERIMENTAL SET UP: | |
| DELAY SIMULATOR TO CHAN. A (MICRO SEC.) | 1.0 |
| DELAY SIMULATOR TO CHAN. B (MICRO SEC.) | 2.0 |
| DELAY CHAN A TO CHAN. B (MICRO SEC.) | 3.0 |
| RUN TIME (MILLI SEC.) | 5000 |

TABLE 5.5 SCENARIO FOR BUS EMULATOR / SIMULATION COMPARISON

SIMULATION/EMULATION COMPARISON

| PARAMETER | SIMULATION | EMULATION |
|---|---|---|
| MESSAGES SENT | 98 | 98 |
| MESSAGES COMPLETE | 98 | 98 |
| MESSAGES CANCELED | 0 | 0 |
| MESSAGES LEFT | 0 | 0 |
| TOTAL BACKOFFS | 185 | 190 |
| MEAN TIME DELAY | 5.955 M.S. | 4.494 M.S. |

| ATTEMPTS | NO. | DELAY | NO. | DELAY |
|---|---|---|---|---|
| 1 | 44 | 1.23 | 11 | 1.65 |
| 2 | 24 | 1.78 | 53 | 2.13 |
| 3 | 9 | 3.16 | 13 | 3.98 |
| 4 | 6 | 4.13 | 6 | 5.11 |
| 5 | 1 | 5.91 | 7 | 4.78 |
| 6 | 0 | – | 2 | 6.39 |
| 7 | 3 | 7.71 | 1 | 5.94 |
| 8 | 2 | 9.89 | 1 | 12.40 |
| 9 | 2 | 16.09 | 0 | – |
| 10 | 2 | 27.29 | 2 | 18.78 |
| 11 | 3 | 51.52 | 0 | – |
| 12 | 1 | 45.48 | 1 | 47.18 |
| 13 | 0 | – | 0 | – |
| 14 | 0 | – | 1 | 77.32 |
| 15 | 1 | 97.70 | 0 | – |
| 16 | 0 | – | 0 | – |

TABLE 5.6 SUMMARY OF EMULATOR / SIMULATION COMPARISON

were completed on the first attempt whereas in the emulator, over half of the messages required 2 attempts. This difference may be caused by the fact that the computer simulation is basically synchronous in nature while the bus emulator is basically asynchronous in nature.

## REFERENCES

[FRAN 81] W. R. Franta and J. Chlamtac, Local Networks, Lexington, Mass: Lexington Books, 1981.

[MOKH 82] N. Mokhoff, "Business Communications", IEEE Spectrum, January, 1982, pp. 43-44.

[O'REI 82A] P.J.P. O'Reilly, J. L. Hammond, J. H. Schlag and D. N. Murray, "Design of an Emulation Facility for Performance Studies of CSMA-Type Local Networks", Proc. 7th Conf. on Local Computer Networks, October 1982, pp.

[O'REI 82B] P.J.P. O'Reilly and J. L. Hammond, "An Efficient Algorithm for Generating the Busy/Idle Periods of a Channel Using CSMA and Loaded by an Arbitrary Number of Stations", Proc. COMPCON Fall 82, Washington DC, September 1982, pp. 427-436.

[SHOC 80] J. F. Shoch and J. A. Hupp, "Measured Performance of an Ethernet Local Network", Communications of the ACM, Vol. 23, No. 12, December 1980, pp. 711-721.

[TANE 81] A. S. Tanenbaum, Computer Networks, Englewood Cliffs, NJ: Prentice-Hall, 1981.

## APPENDIX A

"DESIGN OF AN EMULATOR FACILITY FOR PERFORMANCE STUDIES

OF CSMA - TYPE LOCAL NETWORKS"

A PAPER PRESENTED AT THE 7 TH CONFERENCE ON LOCAL NETWORKS,

MINNEAPOLIS MINNESOTA, OCTOBER 11-13, 1982

# DESIGN OF AN EMULATION FACILITY FOR PERFORMANCE STUDIES OF CSMA-TYPE LOCAL NETWORKS*

Peter J. P. O'Reilly, Joseph L. Hammond, Jay H. Schlag
School of Electrical Engineering, Georgia Institute of Technology, Atlanta, GA.

Dale N. Murray, Advanced Information Systems Division,
Army Institute for Research in Management Information and Computer
Sciences, US Army Computer Systems Command, Atlanta, GA.

## ABSTRACT

The paper discusses the design and preliminary evaluation of a facility for emulating baseband or broadband CSMA-type local networks for communications assessment of distributed processing systems. The facility is being constructed to meet a need for a cost-effective method for studying the performance and protocol interactions of local networks loaded with any number of stations, even up to the order of several thousand.

The facility consists of two physical stations, with high levels of protocols implemented and active, coupled into an emulated network. The emulation facility uses an analytical model to generate the effect of traffic from any number of background stations. A control program allows the physical stations to interact with the emulated background stations so that the former behave as if they are connected to an actual network. In operation, the only transfer of data is into and out of a memory shared by the two physical stations. Thus high speed serial bus operation can be emulated with low speed equipment.

A key element of the facility is the background traffic generator. The proper operation of this generator is verified with studies discussed in the paper.

Keywords: Local Computer Networks, Local Area Networks, Analytical Model for LCNs, Performance Monitoring of LCN, LCN Emulation, LCN Simulation, ETHERNET Performance Characteristics, Communications Assessment Tool for Distributed Processing, Distributed Processing Communications Test-bed Facility.

*This research was done for the Department of Defense under contract number DAAG29-80-K-0009 with the U. S. Army Research Office for the ARMY INSTITUTE FOR RESEARCH IN MANAGEMENT INFORMATION AND COMPUTER SCIENCES. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies, either expressed or implied of the United States government.

## 1. INTRODUCTION

Local computer networks seem destined to play a role of increasing importance in distributed processing and office automation. Within the class of local computer networks, the bus topology, using some form of carrier sense multiple access (CSMA) both baseband and broadband, has a number of advantages in terms of reliability, cost, and ability to reconfigure. This paper describes a cost-effective facility for performance studies of such networks.

The general structure of a CSMA network is shown in Figure 1 using the terminology of Net One

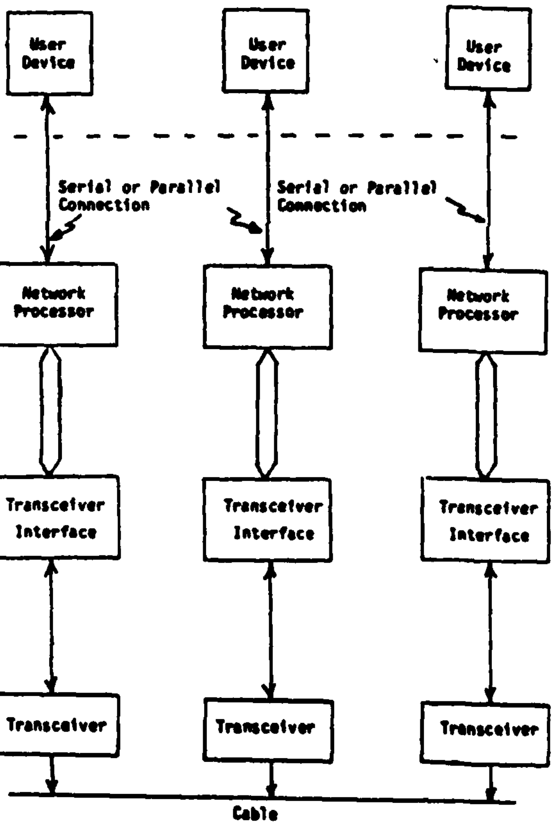


Figure 1. General Structure of a Class of Contention-Type Local Computer Networks (Boxes are identified with the terminology used by Net One.)

[1]. User devices are coupled to a coaxial cable at a number of station locations along its length. Each station has three major components: the network processor, the transceiver interface, and the transceiver. The bandwidth of the cable is such as to allow operation in the megabit per second range; for example, Mokhoff [2] discusses representative local area networks which operate at data rates from 0.8 to 50 megabits per second. Using such high data rates allows a CSMA-type network to support a large number of stations. Ethernet, as an illustration, claims the ability to support over one thousand stations.

The objective of the facility to be described is to provide a cost-effective means for studying the overall performance (as contrasted to merely the performance of the access protocols), of a variety of CSMA networks coupling appropriate user devices under realistic conditions. In order to accomplish the objective, all protocol levels up to the user-to-user level, and of course including the access protocol, must be a part of the study.

Desired results from the facility are such performance relationships as delay versus throughput, and response time versus load for station pairs communicating in the presence of varying amounts of "background" traffic due to other stations competing for the network.

## 2. PREVIOUS PERFORMANCE STUDIES

Most of the performance studies reported in the literature examine the characteristics of only the access protocols, without considering the higher level protocols necessary for user-to-user communication. Studies of this sort for ALOHA-like contention mechanisms, including pure ALOHA, slotted ALOHA, non-persistent CSMA and various p-persistent CSMA schemes, are summarized by Kleinrock [3]. More recently, detailed analytical studies of CSMA/CD protocols have been done by Tobagi and Hunt [4], by Franta and Bilodeau [5], and by Lam [6]. Lam's paper contains simulation curves for comparison to the analytic results, and all three papers give a variety of curves showing the relationship of delay and throughput to other system parameters.

Typical simulation studies of access protocols are those by: Tokoru and Tamaru [7] for Acknowledging Ethernet, Franta and Bilodeau [5] for priority CSMA, Hughes and Li [18] for Ethernet, and Almes and Lazowski [19] for what are termed Ethernet-like networks.

In the only work found which reports measured results, Shoch and Hupp [8], [17], give limited data on channel utilization versus number of hosts in two sets of curves using measured data from an experimental Ethernet.

The most extensive simulation studies reported in the literature to date have been done by Yeh [9], Watson [10] [11], and Donnelley and Yeh [12]. These authors in the related papers cited study HYPERchannel and, in a part of the study [11], compare it to Ethernet. The simulations performed by these authors include some higher level protocols as well as the basic access protocols. The emphasis in these studies is to obtain curves of total network throughput and average delay versus load for numbers of stations varying from two or three to twenty. HYPERchannel uses a hybrid access mechanism involving some aspects of central control. Although central control strategies are not a part of the present study, the papers are mentioned because of the overlap into areas of interest.

In summary it seems accurate to say that, although performance of basic access protocols have been studied to some extent, only a very limited amount of data is available on user-to-user performance, either from measured data or from simulation studies.

## 3. NEED FOR FACILITY

The survey of previous performance studies, documented in Section 2, indicates a need to study the performance of user-to-user links through CSMA-type local networks for both baseband and broadband systems. Such studies could obviously be carried out using the brute force approach of building a test bed with a number of different contention networks deployed. Each network would have to be loaded with stations producing prescribed traffic and the number of required stations could be in the hundreds to achieve a full range of load conditions. This approach, using a brute force test facility, is not cost-effective.

An alternate approach would be to study the networks in question with a simulation model. The straight-forward approach to simulating a CSMA-type local network would use a discrete event model, perhaps programmed using a simulation language such as GPSS or SIMULA (Franta [13]). Such an approach is feasible, but clearly the number of events which must be processed will increase in proportion to the number of stations connected to the network. Experience has shown that as the number of stations in the simulated network becomes forty or so, the cost of computer time becomes significant. The cost of a simulation with on the order of one thousand stations, and including higher level functions, would be prohibitive.

In order to carry out cost-effective studies of user-to-user characteristics for local networks, a different approach to those of a brute force test bed, or a discrete event-type simulation model must be found. Such an approach is the subject of the present paper.

## 4. OVERALL DESIGN OF THE EMULATION FACILITY

### Basic Approach

Consideration of the shortcomings of the two methods discussed above shows that both are subject to costs which increase in proportion to the number of stations connected to the cable, even though attention is to some extent focused on only the two stations which are communicating. This realization leads to a structure for the emulation facility which isolates two communicating stations from all of the other stations which, taken together, comprise a "background" type of load for the cable. The two communicating stations are implemented in physical hardware and software while the background traffic is generated artificially using a model and the specifications of the commercial LCN to be emulated, as shown in Figure 2. With this approach, the two physical stations operate essentially as if they are connected to a real network, while use of the artificial background makes it possible to emulate arbitrarily large loads merely by changing constants in a background computer program.

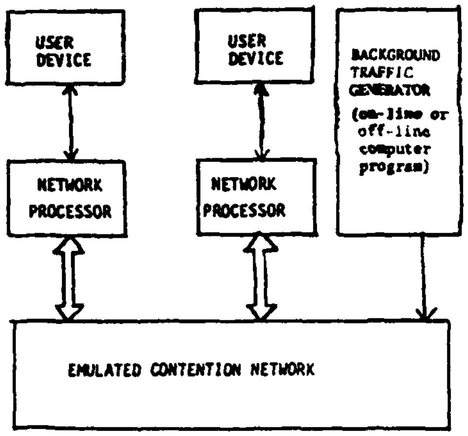

USER DEVICE
USER DEVICE
BACKGROUND TRAFFIC GENERATOR
(on-line or off-line computer program)
NETWORK PROCESSOR
NETWORK PROCESSOR
EMULATED CONTENTION NETWORK

Figure 2. Logical Diagram of Emulated Contention-Type Local Network

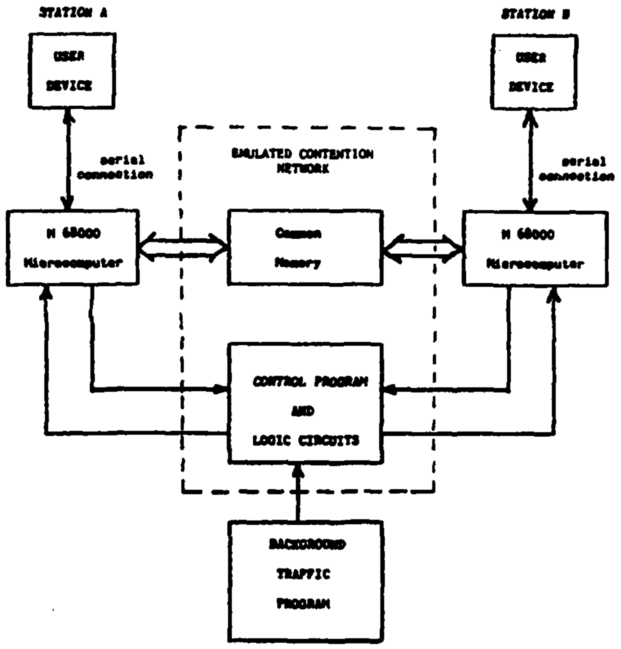


Figure 3. Physical Diagram of Emulation Facility

The operation of the emulation facility can be explained in more detail by referring to Figure 3. Stations A and B are implemented in hardware and software much the same as in an actual network. The M68000 microcomputers carry out the tasks of the network processors in Figure 2. In operation, a packet of data from a user at station A, which is to be transferred to station B across the network, is stored by station A in the common memory. When the control program finds the emulated cable is free of background traffic, station B is allowed to read the packet out of memory to the user at station B after a time delay which accounts for transmission time and propagation delay in the actual network.

Since actual data movement, emulating flow through the network, is only into and out of the common memory, it is possible to emulate large bit rates without using a high-speed serial link. The control program, using inputs from the background traffic program and the logic circuits, operates as a switch which determines when data can be transferred from station A to station B and vice versa.

## Design of Control Program and Logic Circuits

In the actual network, the network processor at each station supports several software programs which are run in real time to control the following functions: frame input, frame output, deference, frame transmission, and frame reception. The "deference" program and the "frame transmit" program require inputs for "channel busy" and "collision detection", which are obtained in the actual network by high speed circuits which monitor the channel. In the emulation facility no physical channel is used and therefore "channel busy" and "collision detection" flags must be generated artificially.

Only two extensive changes in the software of the two physical stations in the emulation facility are necessary due to the modified operation of transferring packets into and out of memory, instead of over a physical channel: one wait time, determined by packet length and actual channel bit rate, is incorporated into the frame transmit program between the point at which the "transmission begins" and the point at which the "transmission" terminates; and a second wait time, equal to the propagation time plus time for the receive process in the actual system, is incorporated into the frame receive program to delay the termination of this process.

The logic for controlling transmissions from stations A and B and for generating channel busy and collision detection flags requires the processing of three types of physical signals: a signal, $s_A(t)$, indicating that station A is transmitting; a similar signal, $s_B(t)$, for station B; and a signal, B(t), indicating that a background station is transmitting. Figure 4 gives typical waveforms for the three signals and a waveform indicating when station A, for example, can use the channel. Previous work has assumed that B(t) is independent of $s_A(t)$, however ways of making B(t) interactive are being investigated.

The "station transmitting" signals are generated in a straightforward manner at each station. The signal indicating that a background station is transmitting is derived by a simple logic

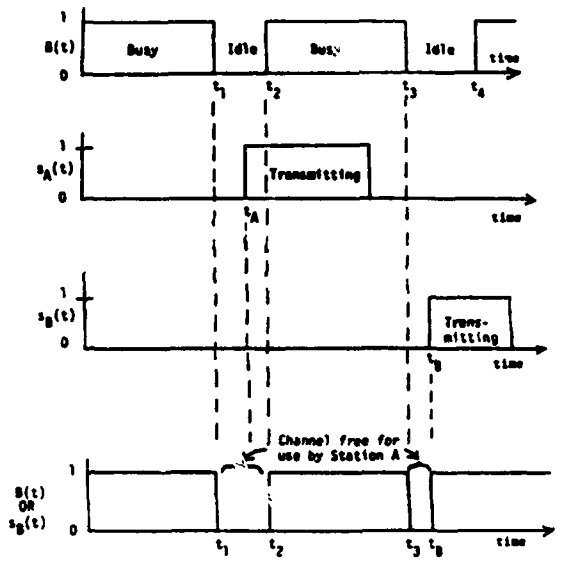
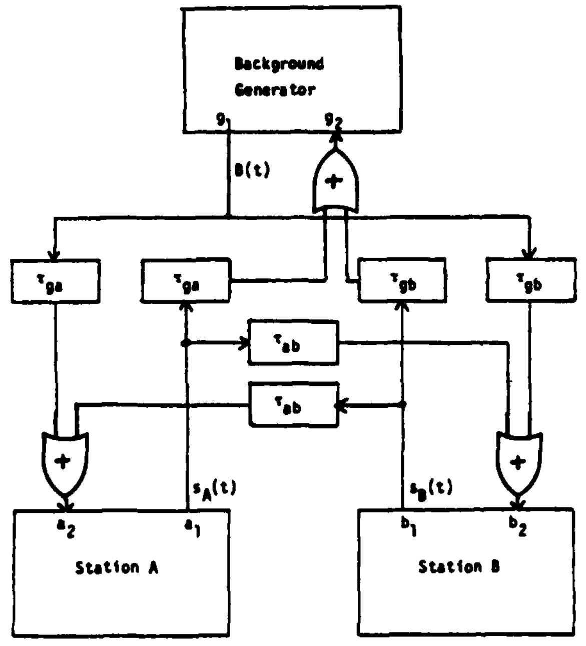


Figure 4. Typical Waveforms for Signals Used in the Logic Circuits of the Control Program.



Deference Logic

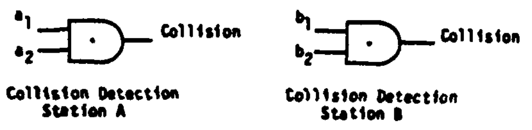


Figure 5. Logic Circuits for Deference and Collision Detection.

circuit from a stored sequence of numbers giving the length of background busy and free intervals. The sequence of numbers for the background is generated, off-line or on-line, by an algorithm to be discussed in Section 5 of the paper.

Figure 5 shows a logic circuit for generating signals to cause stations A and B (and the background generator if so desired) to defer when transmissions in progress from other stations are detected at the station in question. A "high" input at terminal $a_2$, for example, causes station A to abort or defer transmission. Such a signal can also be used as a "channel busy" flag for station A. A high input at $g_2$ inhibits the traffic generation program if so desired.

Note that the logic circuit for deference includes delays $\tau_{ab}$, $\tau_{ga}$, and $\tau_{gb}$, which account for the appropriate propagation delays between stations A, B and an "average" background station. The topology to be emulated and the interstation distances involved decide the settings of these delays.

Collision detection is accomplished by combining the signals $a_1$, $a_2$ or $b_1$, $b_2$ in an AND circuit. At station A, for example, if the signal at $a_1$ is high at the same time that the signal at $a_2$ is high, an impulsive output results from the AND circuit before the $a_2$ input causes station A to abort. Such a signal indicates that station A has observed more than one signal on the channel, and hence a collision has occurred. The impulsive output can be used to actuate a circuit which maintains the "collision detection" signal over several slot times, if this is appropriate for the network being emulated.

## Instrumentation

The emulation facility is to be used to study the performance of various emulated networks. To accomplish this a number of quantities must be monitored in the course of operation.

The parameters of the background generator are determined as a part of the basic algorithm and no difficulty has been experienced in recording or processing this data.

For the actual stations A and B, several measurements must be made. For example, such quantities as the starting time of the first bit of a message and the time of receipt of the last bit of a message must be measured. Similar measurements must be made for packets, since they will not typically be of constant length. Packet lengths must also be measured as packets are constructed.

With respect to channel operation, the number of collisions, the numbers in various queues and possible excess collisions are quantities to be observed. Details of these measurements and circuits will not be discussed in this paper.

## 5. GENERATION OF BACKGROUND TRAFFIC

The design of an algorithm for generating the duration of busy and idle periods for the background traffic represents a significant problem if the system is to accurately represent a real CSMA network. The theoretical aspects of this problem are discussed in detail in another paper by two of the authors [14].

In the next subsection, the algorithm used with the emulation facility is discussed qualitatively and some of the equations defining the algorithm are given.

### Algorithm for Background Traffic

The background traffic algorithm produces the data for a signal, B(t), consisting of busy periods of random duration interspersed with idle periods also of random duration. The statistical properties of the busy/idle periods are determined by the number of stations on the emulated bus, the characteristics of the individual station loads, and of the protocols used - both the specific access and higher level protocols. The final output of the background generation is a two state stochastic process with alternating busy and idle periods.

The traffic algorithm is based on a set of equations which describe the dynamic behavior of the CSMA/CD network. Following the approach used in [15] and [8], probabilistic arguments and some results from the theory of regenerative processes are used to develop these equations.

In developing the equations, it is assumed that the time axis is divided into slots, with the length of each slot taken to be the maximum one-way propagation delay along the part of the bus used by the stations contributing to the background traffic. All time periods are taken to be an integer number of slots, and transmissions are assumed to start only at the beginning of a slot, as for a slotted 1-persistent CSMA/CD protocol.

A time interval of network operation consists of successive periods of successful transmissions and contention intervals with idle slots interspersed, as illustrated in Figure 6(a). Many protocols cause a station to defer for one slot, after detecting an idle channel, before attempting a transmission, in order to allow the last bit of a message to reach all stations on the bus. Such slots, during which the channel is effectively still busy, are shown dotted in Figure 6.

Figure 6(b) illustrates busy and idle periods produced by the traffic generator as B(t). As indicated in the figure, the busy periods can consist of a number of successful transmissions and/or contention intervals.

In both parts of Figure 6, arriving packets are denoted with arrows. Two or more arrivals within an idle slot, a successful transmission period, or a contention interval will cause a collision during the next open slot time, as illustrated in the figure.

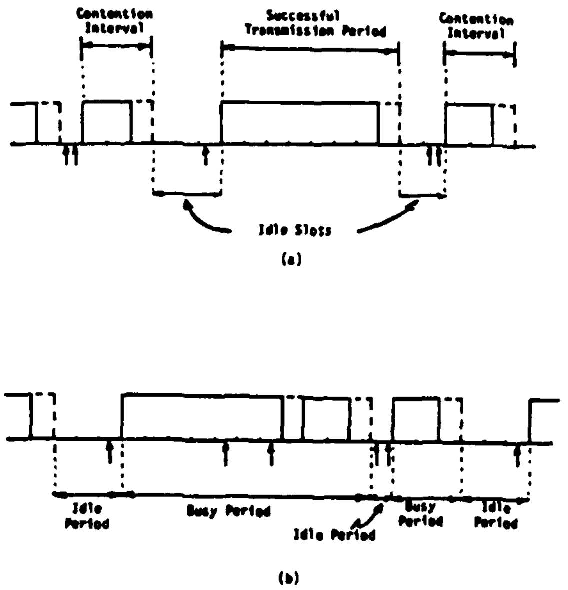


Figure 6. (a) A Segment of Network Time Showing Contention Intervals, Idle Slots and Successful Transmission Periods

(b) A Segment of Network Time Identifying Busy and Idle Periods

Assumptions and notation used in formulating the equations for the traffic algorithm are as follows:

- fixed packet length, L(slots)
- finite number of stations, Q
- identical Poisson arrival process to each station with mean interarrival time, $\gamma$ (slots)
- length of contention interval, N (slots)
- mean backoff time after first collision, $\nu_1$ (slots)
- the arrival process to a particular station is deactivated until the transmission of a packet already at the station is successfully completed.

Stations can be in one of two states: "thinking" or "backlogged." In the thinking state, packets arrive from outside the network in any slot with a probability $\sigma_t$ given by

$$\sigma_t = 1 - \exp(-1/\gamma) .$$

A station is backlogged if its current packet has suffered a collision. Backoff time, for the collided packets, is chosen from an exponential distribution so that in the backlogged state packets may be said to "arrive" (in this case from the store of backed-off packets) at the station with probability $\sigma_b$ given by:

$$\sigma_b = 1 - \exp(-1/\nu)$$

In this expression $\nu$, the mean of the backoff time distribution, is determined by the particular protocol used by the background stations; for example, it may be fixed or may depend in a linear or binary exponential fashion on the average number of collisions per backlogged station.

For each time slot, past events determine if the channel is busy or available. If the channel is busy, the algorithm indexes to the next slot. If the channel is available, three probabilities are computed, namely: the probability of a successful transmission starting, ($p_G$), the probability of a contention starting ($p_C$) and the probability of the slot remaining idle ($p_I$). Using a pseudo-random number generator, one of the three outcomes is selected and, in the case of a successful transmission or a contention, the dynamic variables are updated.

The dynamic variables and the equations for the probabilities of the three outcomes are now summarized.

There are three basic dynamic variables that are potentially updated for each slot: the number of stations with messages in the backlog due to first collisions, $K_A$, those in the backlog due to more than one collision, $K_B$, and the average number of collisions for those stations whose current packet has collided more than once, C. In passing, it should be noted that in general the behavior of individual background stations is not identifiable.

Equations for the three desired probabilities are now given in terms of the more basic probabilities $p_i$ and $q_i$.

$p_i$ is the probability of i new arrivals (from thinking stations) attempting transmissions in the typical slot. This quantity is given by:

$$p_i = \frac{(Q-K)!}{(Q-K-i)!\ i!} (1 - e^{-T/\gamma})^i (e^{-T/\gamma})^{Q-K-i} ,$$

$$i = 0,1,\ldots,Q-K$$

where $K = K_A + K_B$ denotes the total backlog and

$$T = \begin{cases} 1, \text{ if previous slot was idle} \\ L+1, \text{ if a successful transmission is just completed} \\ M+1, \text{ if a contention interval is just completed} \end{cases}$$

The second basic quantity, $q_i$, the probability of i attempted retransmissions occurring in an available slot, is given by:

$$q_i = \sum_{j=0}^{i} q_j^{(A)} q_{i-j}^{(B)} , \quad i = 0,1,\ldots,K .$$

In this expression $q_i^{(A)}$ and $q_i^{(B)}$ are given by:

$$q_i^{(A)} = \frac{K_A!}{(K_A - i)!\,i!} (1 - e^{-T/\nu_1})^i (e^{-T/\nu_1})^{K_A-i} ,$$

$$i = 0,1,\ldots,K_A$$

and

$$q_i^{(B)} = \frac{K_B!}{(K_B - i)!\,i!} (1 - e^{-T/\nu})^i (e^{-T/\nu})^{K_B-i} ,$$

$$i = 0,1,\ldots,K_B .$$

The two quantities $q_i^{(A)}$ and $q_i^{(B)}$ are respectively the probability of i retransmissions arriving from stations with one collision and the probability of i retransmissions arriving from stations with more than one collision. The quantity $\nu$ in the equation for $q_i^{(B)}$ is a function of the backoff algorithm corresponding to a specific protocol. For the emulation of Ethernet-like protocols, $\nu$ is given by:

$$\nu = \nu_1 2^{C-1}$$

The desired probabilities $p_I$, $p_G$, and $p_C$ can now be expressed as:

$$p_I = p_0 q_0$$

$$p_G = p_1 q_0 + p_0 q_1$$

$$p_C = 1 - p_I - p_G$$

A pseudo-random number generator producing X from a uniform distribution as (0,1), is used to decide between the three possibilities according to the following rule:

if $X < p_G$ a successful transmission is initiated

if $p_G < X < (p_G + p_C)$ a collision occurs

if $X > (p_G + p_C)$ the slot remains idle.

Possible outcomes and their relation to the three probabilities are sketched in Figure 7.

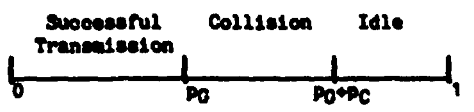


Figure 7. Identification of Ranges for the Variable X.

In a similar manner, other probabilities are calculated and further decisions are made: probability of a retransmission being involved given that a successful transmission is initiated; probability that the backlog $K_B$ is reduced given that a retransmission is involved in the successful transmission; probability of having 0,1,..Q-K new arrivals involved in a collision, of having 0,1,..$K_A$ and of having 0,1,..$K_B$ retransmissions involved in a collision. Using similar random numbers, decisions are made which allow the updating of $K_A$, $K_B$, and C. The above probabilities and discussions illustrate the basic approach used. A number of other similar probabilities must be computed in order to complete the updating. The equations are discussed in detail in the forthcoming paper [14].

## Verification of the Algorithm

The algorithm has been implemented in FORTRAN code and to verify its effectiveness a number of checks have been performed: against the results of Shoch and Hupp [8] for an experimental Ethernet; against the results of an Acknowledging Ethernet; against the results of an in-house discrete event simulation for a small number of stations; and, finally, with an analytic study of the basic equations.

### Verification of the Algorithm with Published Data:

To test the traffic generation algorithm, it can be run as a part of a simulation providing periodic outputs on the number of successful transmissions, collisions, number of stations in the backlog, etc. From this data various overall network performance measures such as "throughput"* can be determined. By considering one or more typical stations as external to the background traffic, delay characteristics can also be studied.

Shoch and Hupp, in the reference cited, give limited data on an experimental Ethernet. The network is operated under high load conditions using a 550 meter bus loaded so that each station accounts for ten percent of the load; i.e., the mean interarrival time was chosen so that the output from each station would occupy the channel for ten percent of the time if no other stations were allowed to transmit.

The results for packets of 4096 bits using the algorithm are compared: to measured results of Shoch and Hupp [17], to ideal throughput response, and to the results of an in-house discrete event simulation in Figure 8 and Table I. As can be noted, results from the algorithm and from the discrete event simulation correspond very closely. A comparison with the Shoch and Hupp data shows more discrepancy but still a good agreement. The differences with the Shoch and Hupp data can probably be accounted for by the fact that Shoch and Hupp do not make clear the condition of their study or how the load was adjusted to give 10% load per station.

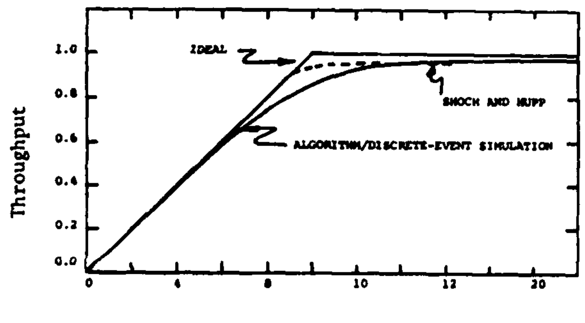


Figure 8. Throughput versus Number of Stations for Packet of Length 4096 Bits.

| Number of Stations | Ideal | Shoch & Hupp | Discrete Event Simulation | Algorithm |
|---|---|---|---|---|
| 5 | 0.5 | 0.50 | 0.478 | 0.482 |
| 8 | 0.8 | 0.80 | 0.729 | 0.735 |
| 10 | 1.0 | 0.94 | 0.864 | 0.856 |
| 12 | 1.0 | 0.96 | 0.924 | 0.922 |
| 15 | 1.0 | 0.96 | 0.961 | 0.959 |
| 20 | 1.0 | -- | 0.976 | 0.971 |
| 100 | 1.0 | -- | --- | 0.977 |

Table I. Throughput versus Number of Stations with each Station generating 10% load.

The results from use of the algorithm are also compared to the simulation results of Tokoro and Tamaru [7] for Acknowledging Ethernet, although their work uses normally distributed interarrival times rather than the, more realistic, Poisson distributed times on which the algorithm is based. The results are shown in Figure 9 for a bit rate of 1 Mbps. The comparison is qualitatively good and discrepancies are considered to be adequately accounted for by the difference in arrival distributions.

### Verification of the Algorithm Using an Analytic Technique:

The background generation algorithm, discussed above, consists of a set of equations for updating values of $K_A$, $K_B$, and C on each of a sequence of time slots. These quantities are sufficient to determine the probabilities $p_G$, $p_C$, and $p_I$, which determine the state of the channel, and hence make possible the slot-by-slot generation of B(t). The

-------------------

*Throughput is defined as the ratio of channel time spent transmitting good packets to total time. Defined in this way, throughput and channel utilisation are equivalent in the present study. Effective transmission rate is obtained as the product of channel capacity with either channel utilisation or throughput.

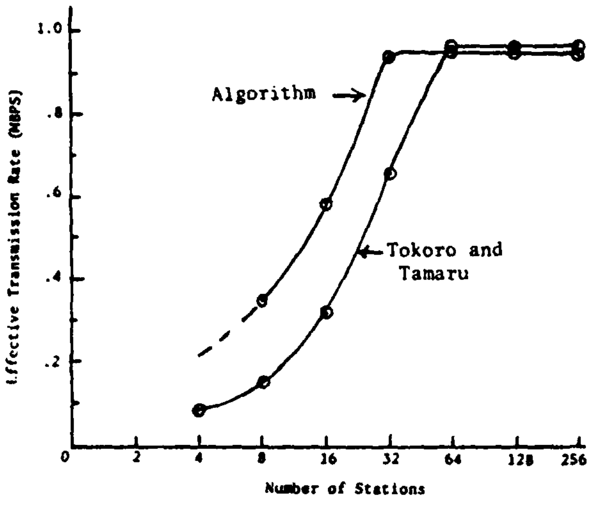


Figure 9. Results from the Background Generator Algorithm Compared to Results from Tokoro and Tamaru for Acknowledging Ethernet.

equations used describe the dynamic behavior of a CSMA/CD network with fewer restrictions than any closed form single expression found in the literature.

A valid analytical check on the equations used, however, can be obtained by reducing the general equations to special cases which have been previously analyzed. One result appropriate for this purpose has been obtained by Kleinrock and Tobagi [16], who find a closed form expression for throughput, S, for a slotted 1-persistent CSMA network without collision detection. Another analysis, by Tobagi and Hunt [4], produces a set of equations which can be solved to give a numerical solution for throughput for a slotted 1-persistent CSMA/CD network under the same restrictive conditions.

To produce the analytic check of the background generation equations, the restrictions used in the Tobagi and Hunt analysis have been applied and an analytic expression for throughput has been obtained. This analytic expression is felt to be a new closed-form expression for the slotted 1-persistent CSMA/CD case, and it gives the same numerical result as Tobagi and Hunt for a particular set of parameter values that they use.

The new result is derived subject to the assumptions that K and $\nu$ are constant with probability 1 and that

$$\lim_{Q\to\infty} \frac{K}{\nu} = 0$$

Details of the analysis are given in the forthcoming paper [14]. The final expression for throughput is

$$S = Lg\, e^{-(N+1)g}\,\{N + 1 - Ne^{-g}\}/D$$

where

$$D = (L+1)g\, e^{-(N+1)g}\,\{N+1 - Ne^{-g}\}$$

$$+ (N+1)\,\{1 - (L+1)g\, e^{-(L+1)g} - e^{-g} + Lge^{-(L+2)g}\}$$

$$+ e^{-(N+1)g} - (L-N)g\, e^{-(N+L+2)g}$$

The result from this equation for S versus g is plotted in Figure 10 for L = 100 and N = 2. This result matches the curve given by Tobagi and Hunt [4] for the same parameter values. The parameter g, defined as the average number of packets offered for transmission in any slot by the non-backlogged stations, is related to the parameters of the background generation algorithm by the equation

$$g = (Q-K)(1 - e^{-1/\gamma})$$

For N = L, and writing G=Lg and a = 1/L, the expression for S reduces to

$$S = \frac{Ge^{-(1+a)G}\,\{1+a-e^{-aG}\}}{(1+a)(1-e^{-aG}) + e^{-(1+a)G}}$$

This expression agrees with that given by Kleinrock and Tobagi in [16].

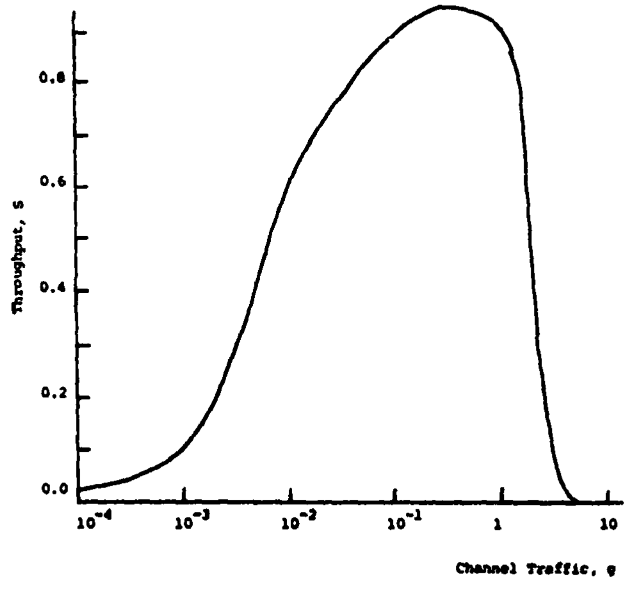


Figure 10. Throughput Versus Channel Traffic

## 6. CONCLUSIONS AND FUTURE PLANS

The studies of the background traffic generator, discussed above, indicate that its operation is satisfactory. Results for throughput versus station load compare adequately well with other data available, given the fact that some approximation is involved in emulating distributed discrete station loads by an equivalent signal derived from average behavior.

To date, the background generator deals mainly with the access protocols of the network. It is planned to introduce further network level protocols into the background generator in the near future, including the queueing of messages at hosts, the partitioning of messages into packets and the transmission of acknowledgements and other control information.

The background traffic generation is judged to be a key part of the emulation facility. Implementation of other parts of the facility is considerably more straightforward and should be completed in the Fall of 1982.

Several types of studies are planned for the completed facility. One class of problems will have to do with determining response times in exchanging data between two stations in the face of varying amounts of background traffic. Such studies are needed for networks operating with both voice and data packets and in other applications where response times are critical.

Another class of problems will deal with determination of throughput versus load for station pairs as a function of various protocols at the data link and higher levels. Such studies will evaluate existing protocols in different combinations and can possibly identify areas for improvement. Studies of this type will aid in assessing the communication requirements and impact on various degrees of distributed data processing system design for future Army needs.

## REFERENCES

[ 1] C. Bass, J. S. Kennedy and J. M. Davidson, "Local Network Gives New Flexibility to Distributed Processing," Electronics, Sept. 25,1980, pp. 114-122.

[ 2] N. Mokhoff, "Business Communications," IEEE Spectrum, January 1982, pp. 43-44.

[ 3] L. Kleinrock, Queueing System, Volume II: Computer Applications, New York: Wiley Interscience, 1976.

[ 4] F. A. Tobagi and V. B. Hunt, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection," Computer Networks, Vol. 4, 1980, pp. 245-259.

[ 5] W. R. Franta and M. B. Bilodeau, "Analysis of a Prioritized CSMA Protocol Based on Staggered Delays," Acta Informatica, Vol. 13, 1980, pp. 299-328.

[ 6] S. S. Lam, "A Carrier-Sense Multiple-Access Protocol for Local Networks," Computer Networks, Vol. 4, 1980, pp. 21-32.

[ 7] M. Tokoro and K. Tamaru, "Acknowledging Ethernet," COMPCON, Fall 1977, pp. 320-325.

[ 8] J. F. Shoch and J. A. Hupp, "Performance of an Ethernet Local Network - a Preliminary Report," COMPCON, Feb. 1980, pp. 318-322.

[ 9] J. W. Yeh, "Simulations of Local Computer Networks - a Case Study," Computer Networks, Vol. 3, 1979, pp. 401-417.

[10] W. B. Watson, "Simulation Study of the Traffic Dependent Performance of a Prioritized CSMA Broadcast Network," Computer Networks, Vol. 3, 1979, pp. 427-434.

[11] W. B. Watson, "Configuration-Dependent Performance of a Prioritized CSMA Broadcast Network," Computer, Vol. 14, Feb. 1981, pp. 51-58.

[12] J. E. Donnelley and J. W. Yeh, "Interaction Between Protocol Levels in a Prioritized CSMA Broadcast Network," Computer Networks, Vol. 3, 1979, pp. 9-23.

[13] W. R. Franta, The Process View of Simulation, New York: North Holland, 1977.

[14] P. J. P. O'Reilly and J. L. Hammond, "An Efficient Algorithm for Generating the Busy/Idle Periods of a Channel Using CSMA and Loaded by an Arbitrary Number of Stations," submitted to appear in the proceedings of COMPCON, Fall 1982.

[15] F. A. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part IV," IEEE Transactions on Communications, Vol. COM-25, pp. 1103-1119, Oct. 1977.

[16] L. Kleinrock and F. A. Tobagi, "Packet Switching in Radio Channels: Part I," IEEE Transactions on Communications, Vol. COM-23, pp. 1400-1416, Dec. 1975.

[17] J. F. Shoch and J. A. Hupp, "Measured Performance of an Ethernet Local Network", CACM, Vol. 23, No. 12, Dec. 1980, pp. 711-721.

[18] H. D. Hughes and L. Li, "A Simulation Model of Ethernet," Technical Report TR #82-008, Department of Computer Science, Michigan State University, 1982.

[19] G. T. Almes and E. D. Lazowski, "The Behavior of Ethernet-Like Computer Communication Networks, " Procedings of the 7th Symposium on Operating Systems Principles, December 1979, pp 66-81.

## APPENDIX B

"AN EFFICIENT ALGORITHM FOR GENERATING THE BUSY/IDLE PERIODS OF A CHANNEL USING CSMA AND LOADED BY AN ARBITARY NUMBER OF STATIONS"

A PAPER PRESENTED AT COMPCON, WASHINGTON D.C.

SEPTEMBER 1982

# AN EFFICIENT ALGORITHM FOR GENERATING THE BUSY/IDLE PERIODS OF A CHANNEL USING CSMA AND LOADED BY AN ARBITRARY NUMBER OF STATIONS

P. J. P. O'Reilly and J. L. Hammond

School of Electrical Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

## ABSTRACT

Many commercial local computer networks which use CSMA/CD access protocols, claim the ability to support hundreds of stations. This paper describes a tool for use in performance studies of such networks when the number of stations is large.

The tool is implemented as an efficient algorithm for approximating the effect of a large number of background stations. The algorithm is based on a probabilistic model for the interaction of the background stations and produces a stochastic sequence of busy/idle periods using standard random number generators. Applications include discrete-event simulation, traffic generation for experiments on an actual network and generation of background for a local network emulation facility.

Verification of the algorithm has been carried out by comparison with experimental results, results from a discrete-event simulation and with theoretical results for a limiting case.

## I. INTRODUCTION

In the last five years, there has been an increasing interest in local area networks and in the Carrier Sense Multiple Access Collision Detection (CSMA/CD) type specifically. Kleinrock[1] discusses CSMA protocols in general while Tanenbaum[2] discusses CSMA and CSMA/CD in the local network context.

Commercial networks of the CSMA/CD type are becoming available and include such networks as Ethernet, Net One and Wang Net[3]. Ethernet is the most widely documented and a detailed description is included as an appendix by Franta and Chlamtac[4]. Most of the commercial networks listed claim the ability to support many hundreds of stations. For example, Ethernet claims to support on the order of one thousand.

This paper is concerned with one aspect of performance studies of CSMA/CD local networks. Most of the performance studies reported in the literature examine primarily the characteristics of access protocols, without considering the higher level protocols necessary for user-to-user communication. The studies typically involve loads from one to possibly a dozen or so stations. The approaches taken fall into the obvious categories of analytical, experimental and simulation.

Detailed analytical studies of CSMA/CD protocols have been carried out by Tobagi and Hunt[5], Franta and Bilodeau[6] and Lam[7]. As with most analytical studies of nontrivial systems, it is necessary to employ approximations to obtain tractable results. Furthermore, in this case, tractable analytical results for complete networks including higher level protocols, seem unlikely. Thus accurate results for the performance of complete networks, including several protocol layers, would seem to require use of either experimental or simulation techniques.

The only work found in the literature reporting experimental results is that of Shoch and Hupp[8,9]. These papers give limited data on channel utilization versus number of stations for an experimental Ethernet. The maximum number of stations used is twenty.

A number of simulation studies have been reported in the last several years. Typical of these studies are the work of Tokoro and Tamaru for Acknowledging Ethernet[10], Franta and Bilodeau[6] for priority CSMA, Hughes and Li[11] for Ethernet and Almes and Lazowska[12] for what are termed Ethernet-like networks. These studies are all directed at low-level protocols and few consider more than a dozen or so stations.

Apparently neither the experimental nor the simulation approach has adequately addressed the performance of CSMA/CD networks loaded with hundreds of stations and including several protocol layers. Reasons for this are apparent: the expense and complexity of a test bed with hundreds of stations for experimental studies, and computing costs, at least directly proportional to modeling detail, for the type of discrete event simulation normally used.

Shoch and Hupp[9] deny the necessary for studies of networks with large number of stations with the claim that one station with heavy traffic is equivalent to many stations with light traffic. This claim, however, has yet to be justified.

This paper describes an efficient algorithm for generating the busy/idle periods of a channel loaded by an arbitrary number of stations. The algorithm can be, potentially, a key element in more efficient performance studies of three types; namely: simulation studies, studies of an artifically loaded physical network with a relatively small number of actual stations, and studies having a combination of physical stations and an emulated network.

Each of these application areas is briefly discussed in the next section of the paper. This discussion is followed by sections detailing the development of the algorithm, verifying the results in a special case with an analytic study, and verifying results with published data. A final section gives a summary and comments on future work.

## II. POTENTIAL APPLICATIONS

As noted above, the algorithm under discussion generates the busy/idle periods for a channel using a CSMA/CD access protocol loaded by an arbitrary number of stations. The algorithm, which can run on-line or off-line on a computer of moderate size, generates, in effect, a sequence of times which delineate the busy and idle periods produced by a collection of stations. The algorithm can be adapted to the access protocol under consideration and the number of stations can be entered as a parameter. In the applications envisioned for the algorithm, the stations of a network are divided into primary stations and background stations. The primary stations which are instrumented for study gain access to the channel in competition with the "background" produced by the algorithm. Details depend to some extent on the particular application *as discussed below.*

### Simulation Studies

In this application the performance of a network is studied with a discrete event simulation. A small selected number of primary stations are modeled in detail including higher level protocols. The event list, which is the heart of the simulation, is then structured to include not only the essential events from the modeled stations but also the busy/idle information from the algorithm, which accounts for the background with which the primary stations compete.

If the background is idle at some time, primary stations can transmit. Happenings at later times can account for possible collisions. If the algorithm indicates that the channel is busy, primary stations must defer in accordance with their access protocols.

In this application, the background algorithm runs concurrently with the remainder of the discrete event simulation program. The algorithm provides an alternative, much more efficient, means of accounting for the bulk of the stations sharing a channel than the normal discrete event representation.

### Artificial Load for a Physical Network

In this application, the algorithm provides the essential information for a background traffic generator for use on an actual network. The algorithm is either run on-line, on a fast computer, or off-line, to generate the random sequence of times delineating the busy/idle periods. The time sequence is then used with a Programmable Pulse Generator, such as that shown in Figure 1, to produce a channel signal V(t). The Pulse Generator box in the diagram of Figure 1 must be designed to produce pulses of the type used by the network under study.

In this application a network installation or test bed, operating with a relatively small number of stations, can approximate the behavior of a fully loaded network.

### Background Traffic for an Emulation Facility

This application is in some respects, similar to the previous one. However, in this case, several physical stations are coupled to an emulation of a physical channel rather than to an actual channel. Such an emulation facility for performance analysis is being constructed. by a group including the authors and is described in a forthcoming paper[13].

The use of the algorithm for generating background traffic is essentially the same as that described for artificial loading of a physical network. For an emulation facility, however, a shaped pulse is not required and the signal B(t) at the output of the flip-flop in Figure 1 serves as the required background signal.

Reference can be made to the paper cited for details of the emulation facility. In this application, as in the the other two, the algorithm provides an efficient means of accounting for the effect of a large number of background stations.

## III. DEVELOPMENT OF THE ALGORITHM

The background traffic algorithm produces the data for a signal consisting of busy periods of random duration interspersed with idle periods also of random duration. The statistical properties of such busy/idle periods are determined by the number of stations on the emulated bus, by the characteristics of the individual station loads, and by the protocols used -the specific access as well as the higher level protocols. The final output of the background generator is a two state stochastic process with alternating busy and idle periods.

The traffic algorithm is based on a set of equations which describe the dynamic behavior of the CSMA/CD network. A flow chart showing the logic structure of the algorithm is given in Figure 2. Following the approach used in[5,14], probabilistic arguments and some results from the theory of regenerative processes are used to develop these equations so as to implement this logic structure.

In developing the equations, it is assumed that the time axis is divided into slots, with the length of each slot taken to be the maximum one-way propagation delay along the part of the bus used by the stations contributing to the background traf-

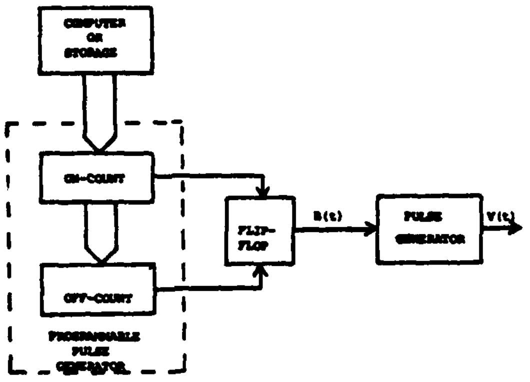


Figure 1. Block Diagram of Generator for Physical Background Signal V(t)

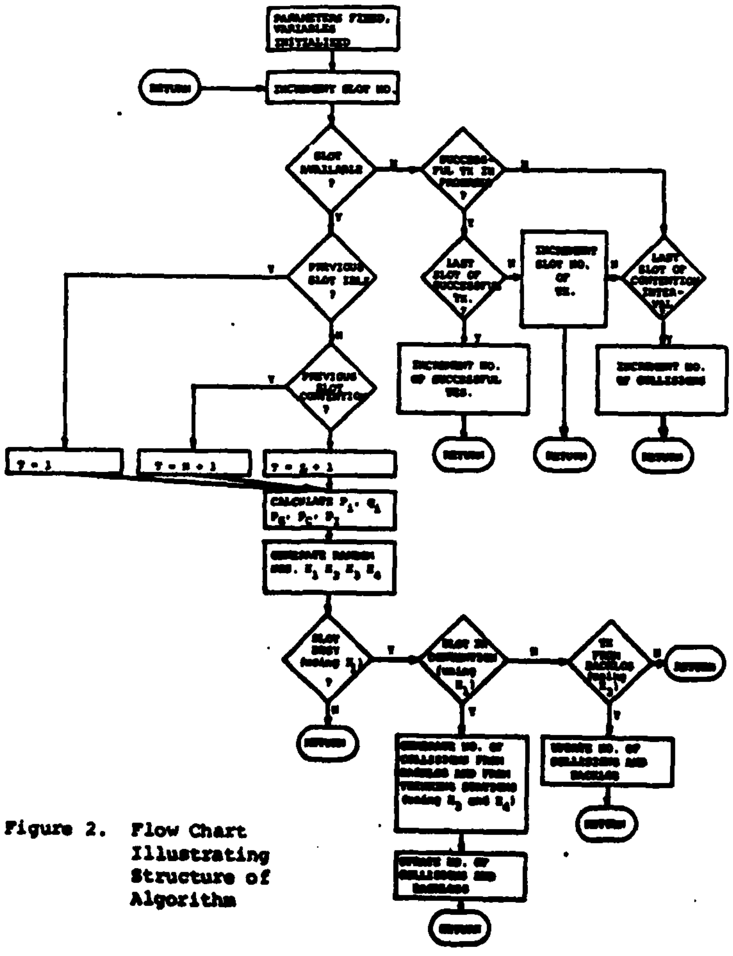


Figure 2. Flow Chart Illustrating Structure of Algorithm

fic. All time periods are taken to be an integer number of slots, and transmissions are assumed to start only at the beginning of a slot. In essence, the analytic approach assumes a slotted 1-persistent CSMA/CD protocol.

A time interval of network operation consists of successive periods of successful transmission and contention intervals with idle slots interspersed, as illustrated in Figure 3(a). Many protocols cause a station to defer for one slot, after detecting an idle channel, before attempting a transmission, in order to allow the last bit of a message to reach all stations on the bus. Such slots, during which the channel is effectively still busy, are shown dotted in Figure 3.

Figure 3(b) illustrates the busy and idle periods produced by the traffic generator. As indicated in the figure, the busy periods can consist of a number of successful transmissions and/or contention intervals.

In both parts of Figure 3, arriving packets are denoted with arrows. Two or more arrivals within an idle slot, a successful transmission period, or a contention interval will cause a collision during the next open slot time, as illustrated in the figure.

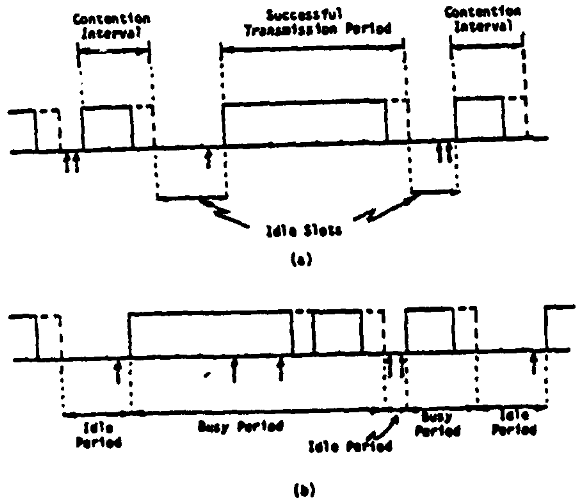


Figure 3. A Segment of Network Time (a) Showing Contention Intervals, Idle Slots and Successful Transmission Periods and (b) Identifying Busy and Idle Periods

Assumptions and notation used in formulating the equations for the traffic algorithm are as follows:

- fixed packet length, L(slots)
- finite number of stations, Q
- identical Poisson arrival process to each station with mean interarrival time, $\gamma$ (slots)
- length of contention interval, N (slots)
- mean backoff time after first collision $\nu_1$ (slots)
- the arrival process to a particular station is deactivated until the transmission of a packet already at the station is successfully completed; only one arrival is permitted to a station during a slot.

Stations can be in one of two states: "thinking" or "backlogged." In the thinking state, a packet arrives to that particular station from outside the network in any slot with a probability $\sigma_t$ given by

$$\sigma_t = 1 - \exp(-1/\gamma) .$$

A station is backlogged if its current packet has suffered a collision. Backoff time, for the collided packets, is chosen from an exponential distribution so that in the backlogged state packets may be said to "arrive" (in this case from the store of backed-off packets) at a station with probability $\sigma_b$ given by

$$\sigma_b = 1 - \exp(-1/\nu)$$

In this expression, $\nu$, the mean of the backoff time distribution, is determined by the particular protocol used by the background stations; for example, it may be fixed or may depend in a linear or binary exponential fashion on the average number of collisions per backlogged station.

For each time slot, past events determine if the channel is busy or available. If the channel is busy, the algorithm indexes to the next slot. If the channel is available, three probabilities are computed, namely: the probability of a successful transmission starting ($p_G$), the probability of a contention starting ($p_C$) and the probability of the slot remaining idle ($p_I$). Using a pseudo-random number generator, one of the three outcomes is selected and, in the case of a successful transmission or a contention, the dynamic variables are updated.

The dynamic variables and the equations for the probabilities of the three outcomes are now summarized.

There are three basic dynamic variables that are potentially updated for each slot: the number of stations with messages in the backlog due to first collisions, $K_A$, those in the backlog due to more than one collision, $K_B$, and the average number of collisions for those stations whose current packet has collided more than once, C. Another dynamic variable, dependent on the other, is the total number of collisions in the current backlog, i.e., $K_A + K_B C$. In passing, it should be noted that in general the behavior of individual background stations is not identifiable.

Equations for the three desired probabilities are now given in terms of the more basic probabilities $p_i$ and $q_i$. The probability of i new arrivals (from thinking stations) attempting transmissions in the typical slot is denoted $p_i$ defined as

$$p_i = \frac{(Q-K)!}{(Q-K-i)!\, i!} (1 - e^{-T/\gamma})^{i} (e^{-T/\gamma})^{Q-K-i} ,$$

$$i = 0,1,\ldots,Q-K$$

where $K = K_A + K_B$ denotes the total backlog and

$$T = \begin{cases} 1, \text{ if the previous slot was idle} \\ L+1, \text{ if a successful transmission is just completed} \\ N+1, \text{ if a contention interval is just completed.} \end{cases}$$

The second basic quantity, $q_i$, the probability of i attempted retransmissions occurring in an available slot, is given by

$$q_i = \sum_{j=0}^{i} q_j^{(A)} q_{i-j}^{(B)} , \quad i = 0,1,\ldots,K$$

In this expression $q_i^{(A)}$ and $q_i^{(B)}$ are given by

$$q_i^{(A)} = \frac{K_A!}{(K_A - i)!\, i!} (1 - e^{-T/\nu_1})^{i} (e^{-T/\nu_1})^{K_A - i} ,$$

$$i = 0,1,\ldots,K_A$$

and

$$q_i^{(B)} = \frac{K_B!}{(K_B - i)!\, i!} (1 - e^{-T/\nu})^{i} (e^{-T/\nu})^{K_B - i} ,$$

$$i = 0,1,\ldots,K_B$$

The two quantities $q_i^{(A)}$ and $q_i^{(B)}$ are respectively the probability of i retransmissions arriving from stations with one collision and the probability of i retransmissions arriving from stations with more than one collision. The quantity $\nu$ in the equation for $q_i^{(B)}$ is a function of the backoff algorithm corresponding to a specific protocol. For the emulation of Ethernet-like protocols, $\nu$ is given by

$$\nu = \nu_1 \, 2^{C-1}$$

The desired probabilities $p_I$, $p_G$, and $p_C$ can now be expressed as

$$p_I = p_0 q_0$$

$$p_G = p_1 q_0 + p_0 q_1$$

$$p_C = 1 - p_I - p_G$$

A pseudo-random number generator producing $X_1$ from a uniform distribution on (0,1), is used to decide between the three possibilities according to the following rule:

*if* $X_1 < p_G$ *a successful transmission is initiated;*
*if* $p_G < X_1 < (p_G + p_C)$ *a collision occurs;*
*if* $X_1 > (p_G + p_C)$ *the slot remains idle.*

Possible outcomes and their relation to the three probabilities are sketched in Figure 4.

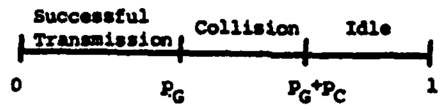


Figure 4. Outcome Intervals for $X_1$.

In a similar manner, other probabilities are calculated and further decisions are made.

A probability G, defined as

G = probability of a retransmission being involved given that a successful transmission occurs,

is introduced to test whether a successful transmission emanates from the backlog or from one of the thinking stations. Clearly,

$$G = p_0 p_1 / p_G$$

If the generated random number, $X_2$, again uniform on (0,1) is less than G, then the backlog is reduced by 1 and the cummulative number of collisions in the backlog is reduced by C.

It is also necessary to evaluate the number of transmission attempts involved in a collision and from whence they originated, i.e., from the backlog or from the group of thinking stations.

Given that a collision occurs, the probability of i or fewer retransmissions being involved in a collision is given by the following set of equations:

$$s_0 = q_0(1 - p_0 - p_1)/p_C$$

$$s_1 = s_0 + q_1(1 - p_0)/p_C$$

$$s_i = s_{i-1} + q_i/p_C , \quad i = 2,\ldots,K$$

If the random number generated, $X_3$, lies between $s_{i-1}$ and $s_i$, then i retransmissions have collided (see Figure 5).

Using a set of similar equations with $p_i$ and $q_i$ interchanged, and another random number, $X_4$, the number of new arrivals involved in the collision are evaluated. This latter number becomes the backlog $K_A$ while the backlog before the collision becomes $K_B$. The total number of collisions is updated by adding the total number of transmission attempts involved in the collision.

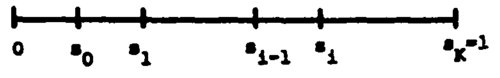


Figure 5. Outcome Intervals for $X_3$.

A feature of many backoff algorithms, such as that of Ethernet, is that the packet is discarded after a specified number of failed transmission attempts. The following analysis shows how this feature is incorporated into the algorithm.

It is assumed, for any station in the backlog, that the number of collisions is geometrically distributed with mean C, that is, the probability of the current packet having had k failed attempts is given by

$$p_k = pq^{k-1}\ , \quad k=1,2,\ldots$$

where

$$p = \frac{1}{C}\ , \quad q = 1 - p\ .$$

Letting M be the maximum number of transmission attempts permitted before cancellation, the probability that any of the K backlogged stations has reached this count is given by

$$1 - (1 - q^M)^K\ .$$

If the generated random number is less than this threshold, then a cancellation is observed; the backlog is reduced by 1 and the total number of collisions is reduced by M.

## IV. ANALYTIC VALIDATION OF THE ALGORITHM

The essence of the algorithm, as discussed in Section III, are the probabilistic expressions, dependent on the current values of $K_A$, $K_B$, and $K_C$, which are evaluated on a sequence of time slots. These quantities are sufficient to determine the probabilities $p_G$, $p_C$, and $p_I$, which determine the state of the channel, and hence make possible the slot-by-slot generation of the busy/idle periods. This set of equations describe the dynamic behavior of a CSMA/CD network with fewer restrictions than any closed form single expression found in the literature.

A valid analytical check on the equations used can be obtained by reducing the general equations, using an infinite population model, to special cases which have been previously analyzed. One result appropriate for this purpose has been obtained by Kleinrock and Tobagi[13], who find a closed form expression for throughput, S, for a slotted 1-persistent CSMA network without collision detection. Another analysis, by Tobagi and Hunt[5], produces a set of equations which can be solved to give a numerical solution for throughput for a slotted 1-persistent CSMA/CD network under the same restrictive conditions.

To produce the analytic check of the background generation equations, the restrictions used in the Tobagi and Hunt analysis have been applied and an analytic expression for throughput has been obtained. This analytic expression is felt to be a new closed form expression for the slotted 1-persistent CSMA/CD case, and it gives the same numerical result as Tobagi and Hunt for a particular set of parameter values that they use. This expression reduces to that of[13] when there is no collision detection.

As discussed in Section III, a number of probabilities are evaluated in each open slot. Clearly these probabilities - of a successful transmission being initiated, of a collision occuring and of the slot remaining idle - are conditional probabilities, dependent not only on the nature of the previous slot but also on the number of stations in the backlog, K, and the mean backoff time, ν. These latter quantities are, of course, random variables.

To adapt the algorithm to the model of[5], it is necessary to make the strong assumption that both these variables are constant with probability 1. It is also assumed that $K_B = K$ and $K_A = 0$.

With these assumptions, the unconditional probabilities $p_g$, $p_c$ and $p_i$ associated respectively with a successful transmission, a contention interval and of a slot remaining idle can be derived from a set of three equations of the form

$$p_g = p_{gc}\, p_c + p_{gg}\, p_g + p_{gi}\, p_i$$

where $p_{gc}$ is the conditional probability of initiating a successful transmission in the slot directly after the conclusion of a contention interval; similar definitions apply to $p_{gi}$, $p_{ic}$, etc. Note that the probabilities $p_{gc}$, $p_{gi}$ and $p_{gg}$ correspond to special cases of $p_G$ in Section III.

Since only two of the three such equations are linearly independent, the equation

$$p_g + p_i + p_c = 1 \tag{1}$$

is also needed to solve for $p_g$, $p_c$ and $p_i$. The solution obtained is:

$$p_g = \frac{p_{gc}(1-p_{ii}) + p_{gi}p_{ic}}{D} \tag{2}$$

$$p_c = \frac{(1-p_{gg})(1-p_{ii}) - p_{gi}p_{ic}}{D} \tag{3}$$

$$p_i = \frac{p_{ic}(1-p_{gg}) + p_{ig}p_{gc}}{D} \tag{4}$$

where the denominator D is such that Eq. (1) is satisfied.

The conditional probabilities in (2)-(4) are essentially the probabilities used in the algorithm and are functions of K, $\nu$, Q, $\gamma$, L and N.

To adapt the model of the algorithm to the simpler infinite population model of[5] a new parameter must be introduced:

g = average number of packets offered for transmission by the non-backlogged (i.e. thinking) states in any slot.

In terms of the model of the algorithm,

$$g = (Q-K)(1 - e^{-1/\gamma})$$

Similarly, another parameter r may be defined as:

r = average number of packets offered for transmission by the backlogged stations in any slot

$$= K(1 - e^{-1/\nu})$$

Assuming an infinite population implies that as $Q \to \infty$, both K and $\nu \to \infty$, while g and r remain finite, i.e.,

$$g = \lim_{Q\to\infty} \frac{Q-K}{\gamma}$$

$$r = \lim_{Q\to\infty} K/\nu \ .$$

(Inherent in such assumptions is that the mean backoff time, $\nu$, increases with Q, i.e., there is no truncation in the backoff algorithm.) For such an infinite model, the conditional probabilities in Eq. (2)-(4) become

$$P_{ii} = e^{-(r+g)}$$

$$P_{gc} = (N+1)(r+g)e^{-(N+1)(r+g)}$$

$$P_{ic} = e^{-(N+1)(r+g)}$$

$$P_{gg} = (L+1)(r+g)e^{-(L+1)(r+g)}$$

$$P_{ig} = e^{-(L+1)(r+g)}$$

The results from the references cited are stated in terms of throughput, S. Using results from renewal theory, S may be written as

$$S = \frac{Lp_g}{(L+1)p_g + (N+1)p_c + p_i} \qquad (5)$$

After doing the necessary substitutions and after some algebraic manipulations, the expression for throughput becomes

$$S = \{L(r+g)e^{-(N+1)(r+g)}\}$$

$$\cdot\{N+1 - Ne^{-(r+g)}\}/M \qquad (6)$$

where

$$M = (L+1)(r+g)e^{-(N+1)(r+g)}\{N+1 - Ne^{-(r+g)}\}$$

$$+ (N+1)\{1 - (L+1)(r+g)e^{-(L+1)(r+g)} - e^{-(r+g)}$$

$$+ L(r+g)e^{-(L+2)(r+g)}\} + e^{-(N+1)(r+g)}$$

$$- (L-N)(r+g)e^{-(N+L+2)(r+g)}$$

In the model considered by Tobagi and Hunt, r=0 and hence Eq. (6) reduces to

$$S = Lg\, e^{-(N+1)g}\{N+1 - Ne^{-g}\}/W \qquad (7)$$

where

$$W = (L+1)g\, e^{-(N+1)}\{N+1 - Ne^{-g}\}$$

$$+ (N+1)\{1 - (L+1)g\, e^{-(L+1)} - e^{-g} + Lge^{-(L+2)g}\}$$

$$+ e^{-(N+1)g} - (L-N)g\, e^{-(N+L+2)g}$$

The relationship between S and g is plotted in Figure 6 for L = 100 and N = 2. This result matches the curve given by Tobagi and Hunt[5] for the same parameter values and thus validates the equations of the algorithm under these restrictions.

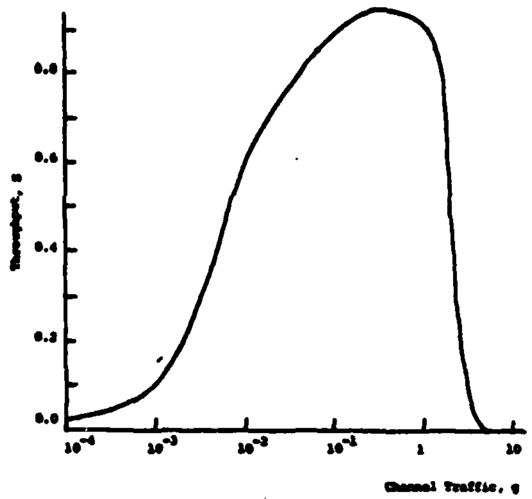


Figure 6. Throughput versus Channel Traffic

For N=L and writing G = Lg and a = 1/L, Eq. (7) reduces further to

$$S = \frac{Ge^{-(1+a)G}\{1 + a - e^{-aG}\}}{(1+a)(1 - e^{-aG}) + e^{-(1+a)G}}$$

This expression agrees with that given by Kleinrock and Tobagi[15].

As a by-product of this validation process, Eq. (6) is a new general closed-form expression for the throughput of a CSMA/CD channel. If it is possible to estimate the average backlog for an actual finite population network under heavy traffic conditions, then the optimum r (and hence $\nu$) for maximum stable throughput can be found from this equation. Unfortunately, this serves only as an upper bound on the throughput and does not help very much in choosing a backoff strategy to achieve such a stable throughput under heavy loads.

## V. VERIFICATION OF THE ALGORITHM

The algorithm has been implemented in FORTRAN code and to verify its effectiveness in emulating network traffic characteristics, a three-way comparison has been made between the results of Shoch and Hupp[8,9] for an experimental Ethernet, a corresponding in-house discrete-event simulation and the algorithm.

To test the traffic generation algorithm, it can be run on its own as a simulation providing periodic outputs on the number of successful transmissions, collisions, number of stations in the backlog, etc. From this data various overall network performance measures such as throughput can be determined. Shoch and Hupp, in the references cited, give limited data on an experimental Ethernet. The network is operated under high load conditions using the following configuration and parameters:

- bus bandwidth: 2.94 Mbps
- bus length: 550 meters
- backoff algorithm: binary exponential backoff, truncated at $2^8$ and with a maximum of 16 transmission attempts; base backoff time ("slot"): 38 μs.
- packet lengths: 256, 512, or 4096 bits.
- number of stations: 5 to 15
- loading per station: 10% per host.
- jam time: 3 μs.

The parameters for both the discrete-event simulation and the algorithm were chosen to be as close as possible to those above; in the case of the algorithm, all time delays must be expressed in units of slots (the maximum one-way propagation delay = 2.38 μs in this case). To enable a direct comparison between the discrete-event simulation and the algorithm, the loading per station was determined in the same way. Shoch does not clearly state how the loading on the Ethernet prototype was adjusted to give 10% load per station.

The arrival process of packets to a station, as described in Section III, was used in both cases. The following brief analysis shows how the mean interarrival time, γ, was evaluated to provide the required loading.

In the case of a single station transmitting on the channel, obviously collisions cannot occur, so that $p_c = 0$.

It can be easily shown that in this case

$$p_g = (1 - e^{-1/\gamma})/(2 - e^{-1/\gamma})$$

$$p_i = 1 - p_g$$

It should be noted that this analysis assumes that a new packet arrival time is not generated until after the interframe delay (one slot time) which follows each successful transmission (see Figure 3). Putting the throughput, S, equal to 0.1 in Eq. (5), the required value for γ may be evaluated:

$$\gamma = \left[\ln\left(\frac{9L-11}{9L-10}\right)\right]^{-1} \text{ slots}$$

where L is the packet length in slots.

For the discrete-event simulation arrival process, a similar analysis in continuous time shows that the mean interarrival time for 10% loading is given in μsecs by

$$\gamma = 9 \text{ (packet transmission time in μs)}$$
$$- \text{ (interframe delay)}$$

The discrete-event simulation assumes that the stations are equally spaced along the bus and so as the number of stations is increased, they become closer together. The programming language used was Pascal, which is very suitable for handling event-driven models. (Although all CSMA/CD network are continuous-time systems, the movement of packets around the network can be expressed in terms of events.)

From Tables 1 and 2, it is evident that excellent agreement as far as throughput is concerned is obtained between the two simulations. Figures 7, 8 and 9 compare, for different packet lengths, a single curve representing the two simulations with the results of Shoch and Hupp and with the ideal throughput response. The difference between the results of the algorithm/discrete-event simulation and Shoch's in the range 6 to 12 is probably due to the differences in the arrival/loading methodologies. For the short packets at high loads, both the simulations tend to give higher throughput than Shoch obtained; however another recent discrete-event simulation of the experimental Ethernet by Hughes and Li[11] tends to agree with the authors' results. It would seem for short packets especially that an operational Ethernet has some secondary delays or other factors which tend to degrade performance at high loads and which need to be modeled. These results seem to provide excellent verification of the operation of the algorithm as far as throughput - load characteristics are concerned.

Tables 1 and 2 give a listing of some of the results obtained for the algorithm and the discrete-event simulation respectively. For each packet size, algorithm results for 100 stations show that stable and fairly constant throughput is obtained under heavy load conditions. The tables also show that, while both approaches produce the same loading effect on the channel, the methods involved are not quite the same: the percentages of collision-free and of cancelled transmission to the number of successful transmissions are quite different, especially for the shorter packets. This discrepancy can possibly be explained by the fact that more collisions are inevitable using the algorithm since it uses the maximum one-way propa-

TABLE 1. Results of the Algorithm for packets of length (a) 512 bits, (b) 1024 bits, and (c) 4096 bits.

| | No. Stations | Throughput | % Transmissions Collision-Free | % Transmissions Cancelled | Average Backlog |
|---|---|---|---|---|---|
| (a) | 5 | 0.482 | 85.0 | 0.0 | 0.1 |
| | 8 | 0.703 | 58.4 | 0.0 | 0.6 |
| | 10 | 0.803 | 34.4 | 0.6 | 2.0 |
| | 12 | 0.839 | 24.2 | 2.0 | 3.5 |
| | 15 | 0.862 | 19.1 | 7.8 | 6.0 |
| | 20 | 0.864 | 15.8 | 15.6 | 9.7 |
| | 100 | 0.913 | 10.7 | 127.8 | 79.9 |
| (b) | 5 | 0.485 | 86.9 | 0.0 | 0.1 |
| | 8 | 0.723 | 59.4 | 0.0 | 0.5 |
| | 10 | 0.823 | 37.6 | 0.0 | 1.5 |
| | 12 | 0.866 | 21.1 | 2.7 | 3.1 |
| | 15 | 0.895 | 15.5 | 8.8 | 4.8 |
| | 20 | 0.911 | 11.5 | 20.2 | 9.0 |
| | 100 | 0.949 | 5.9 | 115.1 | 79.5 |
| (c) | 5 | 0.482 | 89.8 | 0.0 | 0.0 |
| | 8 | 0.735 | 52.0 | 0.0 | 0.5 |
| | 10 | 0.856 | 26.2 | 0.3 | 1.2 |
| | 12 | 0.922 | 16.5 | 3.6 | 2.2 |
| | 15 | 0.950 | 8.5 | 9.5 | 3.8 |
| | 20 | 0.971 | 5.1 | 26.8 | 7.4 |
| | 100 | 0.977 | 3.1 | 109.0 | 79.8 |

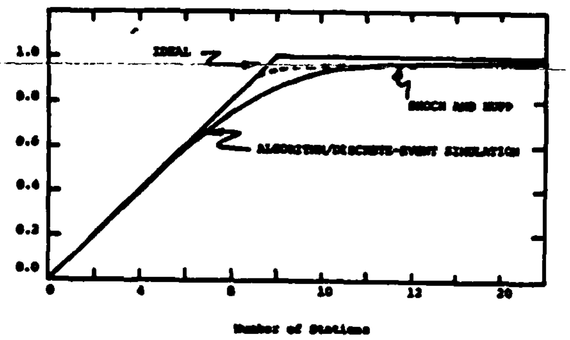


Figure 7. Throughput versus Number of Stations for Packets of Length 4096 Bits

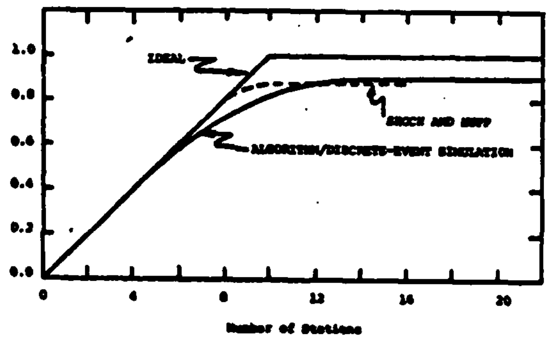


Figure 8. Throughput versus Number of Stations for Packets of Length 1024 Bits

TABLE 2. Results of the Discrete-Event Simulation for Packets of Length (a) 512 bits, (b) 1024 bits, and (c) 4096 bits.

| | No. Stations | Throughput | % Transmissions Collision-Free | % Transmissions Cancelled | Mean* Delay (μs) |
|---|---|---|---|---|---|
| (a) | 5 | 0.480 | 87.0 | 0.0 | 234 |
| | 8 | 0.693 | 62.7 | 0.0 | 410 |
| | 20 | 0.785 | 47.2 | 0.0 | 673 |
| | 12 | 0.817 | 43.0 | 0.0 | 938 |
| | 15 | 0.845 | 36.9 | 0.2 | 1622 |
| | 20 | 0.851 | 29.1 | 0.9 | 2055 |
| (b) | 5 | 0.490 | 88.3 | 0.0 | 466 |
| | 8 | 0.720 | 60.9 | 0.0 | 725 |
| | 10 | 0.823 | 44.1 | 0.0 | 1117 |
| | 12 | 0.877 | 36.0 | 0.1 | 1642 |
| | 15 | 0.893 | 30.5 | 0.7 | 2373 |
| | 20 | 0.902 | 23.1 | 2.2 | 3401 |
| (c) | 5 | 0.478 | 88.5 | 0.0 | 1805 |
| | 8 | 0.729 | 61.4 | 0.0 | 2662 |
| | 10 | 0.864 | 40.0 | 0.1 | 3903 |
| | 12 | 0.926 | 34.9 | 0.3 | 4912 |
| | 15 | 0.961 | 22.3 | 3.6 | 6632 |
| | 20 | 0.976 | 16.1 | 8.9 | 9907 |

*Delays for successful transmissions only.

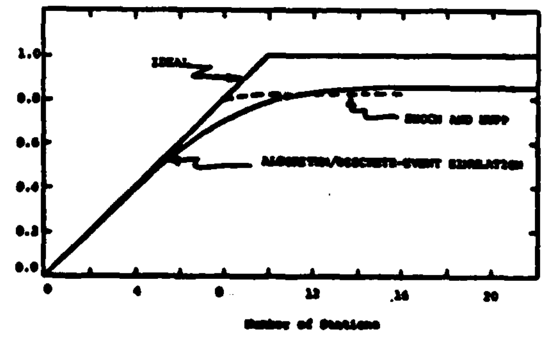


Figure 9. Throughput versus Number of Stations for Packets of Length 512 Bits

gation delay as the effective delay between all stations.

## VI. DISCUSSION

The algorithm described above can generate the busy/idle periods of any CSMA/CD access protocol. Parameters such as number of stations, packet length, length of bus and channel bit rate are set to appropriate values. Parameters accounting for specific details of particular access protocols, such as maximum number of collisions, must also be incorporated.

Using parameters appropriate for Ethernet, comparison with measurements made by Shoch and Hupp and with results of an in-house discrete event simulation show excellent agreement. A theoretical analysis using the equations of the algorithm is also presented and it gives the same results as obtained by other authors for the special cases considered. This analysis and the comparisons noted taken together are felt to provide a complete validation of the algorithm.

For the applications such as those discussed in the paper, the only alternative use of the algorithm is the implementation of a discrete event simulation of the large number of stations in the background. Computing costs, in terms of CPU time and storage, are significantly less for the algorithm. Storage which is at least directly proportional to the number of stations for a discrete-event simulation, is not significantly dependent on the number of stations for the algorithm. Rather than processing a series of events arising from all of the stations, the algorithm deals with a set of equations which are solved using samples from pseudo-random variables generated by the computer. Thus the basic approach used by the algorithm is inherently more efficient.

Several extensions to the study reported in the paper are planned. In one direction, higher level protocols than the access level investigated to date can be added to the algorithm.

In other directions, empirical data can be obtained on computer run times and costs. Measurements can also be made to quantify the length of transients which occur before the steady state is solved.

Finally, external stations can be operated in conjunction with the background generator to obtain delay versus throughput curves which can be compared to results of appropriate discrete-event simulations and, perhaps, experimental data.

## ACKNOWLEDGEMENTS

The work reported was done under contract number DAAG29-80-K-0009 with the U.S. Army Research Office for the Army Institute for Research in Management Information and Computer Sciences.

## REFERENCES

1. L. Kleinrock, Queueing Systems, Volume II: Computer Applications, New York: Wiley Interscience, 1976.
2. A. S. Tanenbaum, Computer Networks, Englewood Cliffs, NJ: Prentice-Hall, 1981.
3. N. Mokhoff, "Business Communications," IEEE Spectrum, January 1982, pp. 43-44.
4. W. R. Franta and I. Chlamtac, Local Networks, Lexington, MA: Lexington Books, 1981.
5. F. A. Tobagi and V. B. Hunt, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection," Computer Networks, Vol. 4, 1980, pp. 245-259.
6. W. R. Franta and M. B. Bilodeau, "Analysis of a Prioritized CSMA Protocol Based on Staggered Delays," Acta Informatica, Vol. 13, 1980, pp. 299-328.
7. S. S. Lam, "A Carrier-Sense Multiple-Access Protocol for Local Networks," Computer Networks, Vol. 4, 1980, pp. 21-32.
8. J. F. Shoch and J. A. Hupp, "Performance of an Ethernet Local Network - Preliminary Report," COMPCON, Feb. 1980, pp. 318-322.
9. J. F. Shoch and J. A. Hupp, "Measured Performance of an Ethernet Local Network," CACM, Vol. 23, No. 12, Dec. 1980, pp. 711-721.
10. M. Tokoro and K. Tamaru, "Acknowledging Ethernet," COMPCON, Fall 1977, pp. 320-325.
11. H. D. Hughes and L. Li, "A Simulation Model of the Ethernet," Technical Report TR #82-008, Dept. of Computer Science, Michigan State University, 1982.
12. G. T. Almes and E. D. Lazowski, "The Behavior of Ethernet-Like Computers Communications Networks," Proc. 7th Symposium on OS Principles, Dec. 1979, pp. 66-81.
13. P. J. P. O'Reilly, J. L. Hammond, J. H. Schlag, and D. N. Murray, "Design of an Emulation Facility for Performance Studies of CSMA-type Local Networks," to be presented at 7th Conference on Local Computers Networks, Minneapolis, MN, Oct. 1982.
14. F. A. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part IV," IEEE Trans. on Communications, Vol. COM-25, Oct. 1977, pp. 1103-1119.
15. L. Kelnrock and F. A. Tobagi, "Packet Switching in Radio Channels: Part I," IEEE Trans. on Communications, Vol. COM-23, pp. 1400-1416, Dec. 1975.

FILME

6 — 8